ANALISA DAN PERANCANGAN SISTEM INFORMASI PERAWATAN PASIEN RUMAH SAKIT IBU DAN ANAK DENGAN MENGGUNAKAN PENDEKATAN BERORIENTASI ASPEK
(STUDI KASUS : RSIA LABUH BARU PEKANBARU)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

Eka Pratiwi Rahmadi

10651004332

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU

2011

ANALISA DAN PERANCANGAN SISTEM INFORMASI PERAWATAN PASIEN RUMAH SAKIT IBU DAN ANAK DENGAN MENGGUNAKAN PENDEKATAN BERORIENTASI ASPEK (STUDI KASUS : RSIA LABUH BARU PEKANBARU)

EKA PRATIWI RAHMADI
10651004332

Tanggal Sidang : 27 Oktober 2011

Periode Wisuda : Februari 2012

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau
Jl. Soebrantas KM 15 No. 155 Pekanbaru

ABSTRAK

Penelitian ini difokuskan pada analisa dan perancangan Sistem Informasi Perawatan Pasien pada RSIA (Rumah Sakit Ibu dan Anak) dengan menggunakan pendekatan berorientasi aspek. Permasalahan yang terdapat pada rumah sakit ini yaitu proses perawatan pasien rawat jalan dan pasien rawat inap, tagihan pasien serta laporan pasien yang belum efektif dan efesien seperti membutuhkan waktu yang lama saat melakukan proses transaksi tersebut. Untuk mengatasi hal tersebut dirancanglah sistem informasi RSIA dengan menggunakan pendekatan berorientasi aspek. Metode penelitian yang digunakan dalam penelitian ini terdiri dari tahap analisa dan perancangan. Tahap analisa dan perancangan dilakukan dengan menggunakan bantuan UML (*Unified Modelling Language*). Pada tahap analisa dan perancangan diperlihatkan perbedaan pendekatan berorientasi objek dan pendekatan berorientasi aspek dan untuk kasus RSIA ini juga dihasilkan *pseudocode* untuk 2 modul yaitu perawatan pasien rawat inap dan perawatan pasien rawat jalan.

Kata kunci : Berorientasi aspek, *Mysql*, PHP, Sistem Informasi, UML.

INFORMATION SYSTEM ANALYSIS AND *DESIGN OF PATIENT CARE HOSPITAL MOTHER AND CHILD APPROACH USING ASPECT ORIENTED (CASE STUDY :* RSIA LABUH BARU PEKANBARU*)*

EKA PRATIWI RAHMADI

10651004332

Date of Final Exam : October 27th 2011

Graduation Period : February th 2012

Informatics Engineering Department
Faculty of Science and Technology
State Islamic University of Sultan Syarif Kasim Riau
Soebrantas *Street* KM 15 No. 155 Pekanbaru

ABSTRACT

This study focused on the analysis and design of Patient Care Information System on RSIA (Mother and Child Hospital) by using aspect oriented approach. Issues contained in this hospital is the process of outpatient care and hospitalization of patients, patient billing and reports of patients who have not been effective and efficient as it takes a long time to process the transaction. To overcome this dirancanglah RSIA information systems using aspect-oriented approach. The research method used in this study consisted of analysis and design phases. Phase analysis and design is done by using the help of UML (Unified Modelling Language). At this stage of analysis and design of object-oriented approach are shown and the differences in aspect-oriented approach and for this case also generated RSIA pseudocode for the two modules namely inpatient care and outpatient care.

Keywords: *aspect-oriented, PHP, Mysql, Information Systems, UML.*

DAFTAR ISI

Halaman

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

RSIA (Rumah Sakit Ibu dan Anak) Labuh Baru yang bertempat di jalan Durian NO. 45 merupakan pelayanan medis yang dilengkapi dengan fasilitas untuk melahirkan, pemeriksaan kehamilan, pemeriksaan ibu dan anak serta berada dibawah pengawasan dokter dan bidan senior. Dalam menjalankan tugas-tugasnya, karyawan RSIA masih menemukan beberapa masalah yang mengakibatkan lamanya pelayanan yang diberikan seperti pencatatan data pasien yang juga membutuhkan waktu yang sangat lama serta pelaporan penyakit pasien yang sulit dilakukan karena penyimpanan seluruh data-data pasien masih dilakukan secara manual, yaitu masih disimpan dalam bentuk kertas folio yang dimasukkan kedalam map pasien. Berdasarkan permasalahan diatas diperlukan sebuah program aplikasi sistem informasi perawatan pasien yang didukung oleh salah satu bahasa pemrograman PHP dengan *database* MySQL karena pemrograman PHP telah mendukung pemodelan visual dengan menggunakan UML yang dapat menentukan objek, kelas serta aspek. UML *(Unified Modelling Language)* merupakan standar dalam menentukan,visualisasi, konstruksi, dan mendokumentasikan *artifact* dari sistem *software*, untuk memodelkan bisnis, dan sistem *nonsoftware* lainnya. Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.

Sehubungan dengan peningkatan pelayanan rumah sakit akan dilakukan secara terus menerus, maka perangkat lunak yang dibutuhkan adalah perangkat lunak yang mudah untuk dikembangkan pada masa yang akan datang. Maka metodologi yang cocok digunakan pada kasus ini adalah metodologi pendekatan berorientasi aspek. Pemrograman berorientasi aspek merupakan paradigma pemrograman yang relatif baru, diperkenalkan sebagai hasil dari penelitian yang dilakukan oleh *Gregor Kiczales* di *Xerox 's Palo Alto Research Center* (PARC). Paradigma ini dikembangkan sebagai salah satu solusi untuk persoalan *separating*

crosscutting concerns pada kode program. Dengan pendekatan pemrograman berorientasi aspek, persoalan didekomposisi menjadi kumpulan kelas (*class*) dan aspek (*aspect*). Kelas mewakili komponen–komponen yang memiliki peran fungsional dalam domain persoalan sistem perangkat lunak yang akan dikembangkan. Sedangkan aspek adalah bagaimana memandang persoalan pemrograman dengan konsep fungsionalitas terhadap suatu kasus.

Metodologi berorientasi aspek berada di posisi yang sama dengan metodologi berorientasi objek yaitu berorientasi konsep. AOP (*Aspect Oriented Programming*) ada untuk melengkapi OOP (*Object Oriented Programming*). AOP menambahkan konsep baru dan sederhana ke dalam OOP untuk lebih mempertajam sifat (*modularity)* yang ada dalam OOP.

Sesuai dengan latar belakang diatas, penulis tertarik untuk mengangkat menjadi sebuah judul penelitian dan pembuatan tugas akhir dengan judul

"ANALISA & PERANCANGAN SISTEM INFORMASI PERAWATAN PASIEN PADA RUMAH SAKIT IBU DAN ANAK DENGAM MENGGUNAKAN PENDEKATAN BERORIENTASI ASPEK".

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah diatas, maka dapat dirumuskan permasalahan yang diangkat yaitu, "Bagaimana menganalisa dan merancang sistem informasi perawatan pasien pada rumah sakit ibu dan anak dengan menggunakan pendekatan berorientasi aspek".

1.3 Batasan Masalah

Dalam analisa dan perancangan sistem perawatan pasien rumah sakit ibu dan anak ini hanya digunakan untuk Rumah Sakit Ibu dan Anak Labuh Baru saja. Batasan masalahnya adalah:

1. Analisa dan perancangan pada sistem informasi perawatan pasien RSIA Labuh Baru ini hanya mengelola pencatatan data pasien rawat inap dan rawat jalan.
2. Analisa dan perancangan pada sistem informasi perawatan pasien RSIA Labuh Baru tidak sampai pada sistem yang akan dibangun.

1.4 Tujuan Dan Manfaat Penelitian

Adapun tujuan dan manfaat yang ingin dicapai dalam pembuatan laporan tugas akhir ini adalah :

1.4.1 Tujuan Penelitian

Merancang dan membangun sebuah Analisa & Perancangan Sistem Informasi Perawatan Pasien pada Rumah Sakit Ibu dan Anak dengan menggunakan metodologi berorientasi aspek.

1.4.2 Manfaat Penelitian

1. Bagi instansi :

   Hasil penelitian ini diharapkan dapat digunakan untuk memonitoring perawatan pasien sehingga terkelola dengan baik dan tetap terkoordinasi.

2. Bagi penulis

   Penelitian ini akan mendukung peneliti dalam mempelajari, menganalisa, dan mengembangkan ilmu-ilmu yang telah diperoleh untuk diterapkan di dunia nyata.

3. Bagi Dunia Akademis

   Bagi pihak lain yang berkepentingan, hasil dari penelitian ini sebagai bahan informasi dan relefansi bagi peneliti/calon peneliti selanjutnya demi pengembangan pengetahuan dan menyempurnakannya kedalam sistem informasi yang lebih kompleks.

1.5 Sistematika Penulisan

Sistematika penulisan Tugas Akhir ini dibagi menjadi beberapa bab, hal ini dimaksudkan agar dapat diketahui tahapan dan batasannya. Adapun sistematikanya sebagai berikut :

BAB I PENDAHULUAN

Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini menjelaskan tentang pembahasan konsep Dasar Sistem, konsep sistem informasi, dan penjelasan pendekatan berorientasi aspek.

BAB III METODOLOGI PENELITIAN

Pada Bab ini akan membahas mengenai metodologi serta langkah-langkah dalam melakukan penelitian dan penyusunan tugas akhir.

BAB IV ANALISA DAN PERANCANGAN

Pada bab ini merupakan pembahasan tentang analisis perangkat lunak, meliputi analisis, analisis masalah, analisis metode, analisis kebutuhan sistem, serta perancangan. Perancangan sistem yang terdiri dari perancangan diagram alir (*flowchart*).

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas implementasi dan pengujian yang dilakukan terhadap Analisa & Perancangan Sistem Informasi Perawatan Pasien dengan menggunakan pendekatan berorientasi aspek atau AOP *(Aspect Oriented Programming)*.

BAB VI PENUTUP

Bab ini berisi kesimpulan yang dihasikan dari pembahasan tentang rancang bangun sistem informasi perawatan pasien RSIA (Rumah Sakit Ibu dan Anak) dengan menggunakan pendekatan berorientasi aspek dan saran sebagai hasil akhir dari penelitian yang telah dilakukan.

BAB II
LANDASAN TEORI

Suatu sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu. Prosedur tersebut adalah suatu urut-urutan operasi tulis-menulis dan biasanya melibatkan beberapa orang di dalam satu atau lebih departemen yang diterpkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi.

2.1 Konsep Dasar Sistem Informasi

Sistem informasi dapat didefenisikan sebagai suatu sistem di dalam organisasi yang merupakan kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur-prosedur dan pengendalian yang ditujukan untuk mendapatkan jalur komunikasi penting memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan yang lainnya terhadap kejadian-kejadian internal dan eksternal yang penting dan menyediakan suatu dasar informasi untuk pengambilan keputusan (Jogiyanto, 1999).

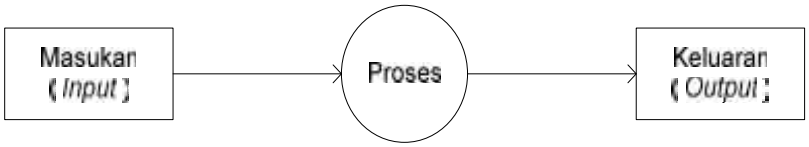
Menurut Kristanto (2003). Sistem Informasi dan Aplikasinya, menyatakan bahwa: "Sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu". Sistem juga merupakan kumpulan elemen-elemen yang saling terkait dan bekerja sama untuk memproses masukkan *(input)* yang ditujukan kepada sistem tersebut dan mengolah masukkan tersebut sampai menghasilkan *(output)* yang di inginkan.

Karateristik sistem terdiri dari:

1. Mempunyai komponen (*componens*)
2. Mempunyai batas (*boundary*)
3. Mempunyai lingkungan (*environment*)
4. Mempunyai penghubung antar muka (*interface*) antar komponen
5. Mempunyai masukan (*input*)
6. Mempunyai Pengolahan (*processing*)
7. Mempunyai keluaran (*output*)

8. Mempunyai sasaran (*objectives*) dan tujuan (*goal*)
9. Mempunyai kendali (*control*)
10. Mempunyai umpan balik (*feed back*)

Model umum sistem adalah terdiri atas masukan (*input*), pengolah (*processing*) dan keluaran (*output*), sebagaimana yang ditunjukkan pada gambar di bawah ini.



Gambar 2.1 Model Umum Suatu Sistem

(Sumber: Jogiyanto, 2000)

1. Masukan (*input*)

   Input adalah semua data yang dimasukkan ke dalam sistem dan selanjutnya menjadi bahan untuk diproses, contohnya saja data transaksi.
2. Proses yang terjadi pada sistem ini proses penyimpanan data dan laporan.

   Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna.
3. Keluaran (*output*)

   Keluaran (*output*) merupakan hasil dari pemrosesan. Keluaran bisa berupa suatu informasi, saran, cetakan laporan, dan sebagainya.

2.2 Komponen Sistem Informasi

Untuk mendukung lancarnya suatu sistem informasi dibutuhkan beberapa komponen yang fungsinya sangat vital di dalam sistem informasi. Secara rinci komponen-komponen sistem informasi dapat dijelaskan sebagai berikut :
(Kristanto,2003) :

1. *Input*

   Input adalah semua data yang dimasukkan ke dalam sistem informasi. Dalam hal ini yang termasuk ke dalam *input* adalah dokumen-dokumen, formulir-formulir dann *file-file*. Dokumen-dokumen tersebut dikumpulkan dan dan dikonfirmasikan ke suatu bentuk sehingga dapat diterima oleh pengolah yang meliputi:

a. Pencatatan
b. Penyimpanan
c. Pengujian
d. Pengkodean

2. Proses

Proses merupakan kumpulan prosedur yang akan memanipulasi input yang kemudian akan disimpan dalam bagian basis data dan seterusnya akan diolah menjadi suatu output yang akan digunakan oleh si penerima. Komponen ini dalam tugasnya akan merubah segala masukan menjadi keluaran yang terdiri dari :

a. Metode dan Prosedur

Metode adalah teknik pengolahan data yang diterapkan pada sistem informasi, sedaangkan prosedur menggambarkan bagaimana manusia sebagai pemakai sistem membuat keputusan.

b. Peralatan Komputer

Komponen pendukung sistem informasi yang termasuk peralatan komputer adalah : monitor, printer, disket dan program komputer. Dalam program komputer terdapat sejumlah instruksi-instruksi yang mengatur kerja dari perangkat keras dan memenuhi fungsi dari sistem informasi komputer.

c. Penyimpanan Data

Berfungsi untuk pemakaian di masa yang akan datang atau pencarian kembali. Media penyimpanan dapat berupa disket, kartu plong, dokumen atau bentuk lainnya.

3. *Output*

Output bisa berupa suatu informasi, saran, cetakan laporan, dan sebagainya.

Komponen ini akan berhubungan langsung dengan pemakai sistem informasi dan merupakan tujuan akhir pembuatan yang dibutuhkan oleh pemakai sistem untuk memantau keberhasilan suatu organisasi.

4. Teknologi

   Teknologi disini merupakan bagian yang berfungsi untuk memasukkan input, mengolah input dan menghasilakan keluaran. Ada tiga bagian dalam teknologi ini yang meliputi perangkat keras, perangkat lunak, dan perangkat manusia. Perangkat keras contohnya : *keyboard, mouse*, dan lain-lain. Perangkat lunak contohnya program untuk mengolah data dan perangkat manusia contohnya analis sistem, programmer, teknisi dan sebagainya.

5. Basis data

   Basis data merupakan kumpulan-kumpulan data yang saling berhubungan satu dengan yang lain yang disimpan dalam perangkat keras komputer dan akan diolah menggunakan perangkat lunak. Basis data sendiri merupakan kumpulan file-file yang mempunyai kaitan antara satu file dengan file yang lain sehingga membentuk satu bangunan data.

6. Kendali

   Kendali dalam hal ini merupakan semua tindakan yang diambil untuk menjaga sistem informasi tersebut agar bisa berjalan dengan lancar dan tidak mengalami gangguan. Komponen ini sangat penting agar sistem secara keseluruhan memiliki validasi dan integritas yang tinggi.

2.3 Sumber Daya Sistem Informasi

Teknologi informasi dapat dikatakan sebuah suatu teknologi yang berhubungan dengan pengolahan data menjadi informasi dan proses penyaluran informasi tersebut dalam batas-batas ruang dan waktu. Komputer adalah satu produk dalam domain teknologi informasi, yang lainnya dalah modem, router,Oracle, Printer, Multimedia dan lain sebagainya.

Sistem informasi merupakan suatu kumpulan dari komponen-komponen dalam suatu organisasi yang berhubungan dengan proses penciptaan dan aliran informasi. Dalam hal ini, teknologi informasi hanya merupakan salah satu komponen kecil saja. Komponen lainnya secara umum adalah proses dan

prosedur, struktur organisasi, SDM (Sumber Daya Manusia), model-model untuk analis, perencanaan, pengendalian dan pembuatan keputusan serta *database*.

Secara umum sistem inforamsi merupakan kombinasi dari orang (*people*), perangkat keras (*hardware*), perangkat lunak (*software*), jarinagn komunikasi (*communications network*) dan sumber data yang dihimpun, ditranformasi, dan mengalami proses pengaliran dalam sumber organisasi.

2.4 Sejarah *Aspect Oriented Programming* (AOP)

Pada tahun 1996, Gregor Kickzales berangkat dari ide *reflection* dalam pemrograman, melontarkan gagasan brilian tentang pemrograman berbasis *aspect* (*aspect-oriented programming*/AOP). Pada tahun 1998-2000, Gregor Kickzales mendapat grant dari DARPA untuk melakukan riset tentang AOP yang terbagi menjadi 2 fase. Fase pertama 1998-1999 lebih banyak tentang AOP itu sendiri sebagai landasan, sementara fase kedua 2000-2002 adalah pengembangan AspectJ sebagai implementasi AOP. Walaupun pada masa awal ide ini boleh dikatakan tidak terlalu banyak mendapat dukungan, namun sejak tahun 2000, melalui usaha tiada henti dan memberikan contoh yang lebih nyata dengan implementasi AspectJ, akhirnya publik pun mulai menunjukkan penerimaannya terhadap gagasan ini. *Project* dan implementasi pemrograman berbasis *aspect* ini pun mulai marak dalam berbagai bahasa. AspectC merupakan *project* implementasi *aspect-oriented* dengan bahasa C/C++, AspectS untuk Smalltalk, AspectNet untuk C# .Net, Pythius untuk Phyton, dan AspectR untuk Ruby. Sementara AspectJ, HyperJ, DemeterJ, Java Aspect Component (JAC dari AOPSYS), Nano, AspectWerkz, dan lain sebagainya, merupakan jajaran nama-nama *project* implementasi *aspect-oriented* untuk Java. Sementara itu JSR 1.5 ‘Tiger’, memasukan ide metadata yang merupakan komplementari dari *aspect-oriented*. Pemrograman berbasis aspek (*aspect-oriented programming*) hadir untuk memotong silang (*crosscut*) permasalahan *scaterring* dan *tangling*. Metodologi ini bukan menggantikan metodologi pemrograman berbasis objek atau pendahulunya, namun hanya merupakan ekstensi pengembangan dari yang ada.

2.4.1 Pengertian *Aspect Oriented Programming* (AOP)

AOP merupakan metode pemograman yang menghadirkan konsep untuk memotong silang (*crosscut*) permasalahan *tangling* dan *scattering*. *Tangling* dalam artian bahwa terdapat hal-hal lain didalam suatu *object* yang sebenarnya tidak berhubungan langsung, namun hanya merupakan *technical concerns* semata. Sedangkan *scattering* lebih kepada pengkodean yang muncul pada banyak tempat, sebagai contoh kode program untuk akses *user password* yang bisa terjadi pada beberapa layer. Kedua hal diatas, telah menyebabkan *source code* menjadi tidak bersih dan tidak independent. Untuk itulah dibutuhkan modul yang dapat melakukan *crosscutting concern* terhadap permasalahan diatas, inilah yang dinamakan aspek. AOP merupakan penyempurna dari metode OOP. Bila kita bandingkan dengan *Object-Oriented Programming* (OOP), *security* dijadikan sebagai sebuah *class*, dan tidak dapat digunakan *crosscutting concern*. Dengan semakin dikenalnya AOP, kemudian muncul Unified Modeling Language (UML) yang menyediakan sintaks grafik untuk membuat model berbasis objek. Contoh *crosscutting concern*:

(*Aspect-Oriented Software Development with Usecase@Team LiB*):

1. *Performance Monitoring*
2. *Logging*
3. *Transaction Management*
4. *Caching*
5. *Security*
6. *Error Handling.*

2.4.2 Komponen *Aspect Oriented Programming* (AOP)

1. *Jointpoint* adalah suatu titik pengeksekusian pada aplikasi, misalnya pada saat pemanggilan method, inisialisasi class, field assignment atau inisialisasi object.
2. *Advice* adalah kode yang akan di eksekusi oleh jointpoint. Ada beberapa macam advice misal: before advice dieksekusi sebelum jointpoint, after advice di eksekusi setelah j*ointpoint*.

3. *Poincut* adalah kumpulan beberapa jointpoint yang digunakan untuk mendefinisikan kapan advice akan dijalankan.
4. *Aspect* adalah kombinasi antara advice dan poincut. Hasil kombinasi ini yang akan menghasilkan logic yang harus di eksekusi oleh aplikasi,
5. *Target* adalah object yang dimodifikasi oleh AOP. Misalkan ada method "setting" yang ada akan ditambahkan advice. Maka method "setting" disebut target.

Selain komponen di atas AOP juga mengenal istilah *weaving* yaitu adalah proses memasukkan aspect kedalam kode pada suatu aplikasi. Ada yang melakukan *weaving* pada saat compile time, namun ada juga yang melakukan *weaving* pada saat runtime.

2.4.3 Implementasi AOP.

Implementasi AOP secara umum dari suatu aplikasi dapat dibagi menjadi tiga bagian:

1. ***Component* program**

   Bagian aplikasi ini merepresentasikan dekomposisi fungsional. Ini diprogram dalam bahasa komponen, seperti Java atau juga suatu bahasa spesifik aplikasi.

2. ***Aspect* Program**

   Bagian aplikasi ini merepresentasikan *crosscut* dengan unit fungsional yang terdapat dalam komponen program. Ini dapat ditulis dalam suatu bahasa spesifik aspek seperti *AspectJ* atau ditentukan dengan mekanisme konfigurasi dengan pola tertentu seperti JAC.

3. ***Weaving***

   Bagian ini merubah komponen program dan aspek program menjadi sistem yang final. Ini mengintegrasikan instruksi ekstra yang dihasilkan untuk aspek kedalam desain komponen. Ini bisa dilakukan pada *compile-time* (*AspectJ* atau *runtime* (JAC). Di satu sisi, beberapa implementasi AOP menggunakan bahasa pemrograman yang sudah ada untuk komponennya. Sebagai contoh, bahasa komponen untuk AspectJ adalah Java. Ini juga mungkin untuk membuat ekstensi untuk bahasa

pemrograman yang sudah ada seperti C, Smalltalk, atau Ruby. Di sisi lain, beberapa implementasi melibatkan bahasa komponen khususnya dibuat untuk tujuan fungsional dari aplikasi. Pasti lebih baik mengenkapsulasi *concern* fungsional dengan *application specific construct*.

2.4.4 Hubungan *Aspect Oriented* dengan *Object Oriented*:

Sebenarnya AOP bukan merupakan pengganti dari OOP. AOP hanya sebagai penambah dari kekurangan terdapat dalam OOP. Kontribusi utama dari AOP ialah memberikan jalan untuk mengintegrasikan bersama-sama dalam sebuah aspek. Aspek sering digambarkan sebagai *crosscutting concern*. Seperti dikatakan oleh *Gregor kiczales* dalam penelitiannya bahwa "*AOP is about capturing crosscutting structure*". Definisi dari aspek hampir umum dengan kelas. Ketika memodelkan suatu masalah, user sering merepresentasikannya dengan menggunakan objek dan setiap objek berisi data (atribut) dan proses (methode). Hal itu juga terjadi pada aspek, aspek digunakan untuk mengimplementasikan (*security, persistence, logging*, dll) pada aplikasi, fungsi ini sama dengan kebutuhan dan pemrosesan. Dengan menggunakan AOP, aplikasi berisi kelas dan aspek. Suatu aspek berbeda dengan kelas dalam implementasi menggunakan *crosscutting functionality*. *Crosscutting functionality* termasuk prosedur atau *object oriented* paradigma. Memasukan kelas dan aspek kedalam sebuah aplikasi berarti *modularity* dapat terjadi dalam 2 dimensi. Fungsi dasar diimplementasikan oleh kelas (dimensi ini dapat disebut kedalam dimensi struktural), dan *crosscutting functionality* di implementasikan oleh aspek (dimensi ini disebut dengan dimensi operational).

Dengan program berorientasi objek, kita dapat meninjau kelas atau set kelas terkait. Dalam sebuah sistem *Object Oriented*, setiap kelas benar-benar merangkum data dan perilaku konsep tertentu. Dengan AOP, kita tidak bisa mengetahui kelas hanya dengan melihat kode. Kita tidak tahu apakah kode tersebut mungkin baik ditambah dengan saran dari beberapa aspek atau sama sekali diganti dengan saran tersebut. Untuk dapat kode aplikasi, kita harus mampu melihat kode dari setiap kelas serta kode untuk setiap aspek yang mungkin mempengaruhi perilaku kelas. Cara untuk berpikir tentang kebenaran AOP kode

adalah kebalikan dari bagaimana kita mempertimbangkan untuk berorientasi objek (OOP) kode pemrograman.

Masalah kedua yang lebih praktis mengenai adopsi potensi AOP adalah pengembangan alat dan teknik untuk pengujian terutama unit testing. Karena kode dapat diubah oleh suatu aspek, unit test yang berjalan sempurna di kelas dapat berperilaku cukup berbeda ketika kelas ini diintegrasikan menjadi sebuah sistem AOP.

AOP banyak berbeda dengan OOP dalam hal menentukan *crosscutting concern*. Dengan AOP, tiap implementasi *concern* tidak perlu memperhatikan *concern* yang lain. Sebagai contoh modul pemrosesan *credit card* tidak perlu mengerti *concern* yang lain seperti *logging* atau *authentication* dan bagaimana operasinya. Ini merepresentasikan pergeseran paradigma yang besar dari OOP.
Suatu implementasi AOP dapat diberlakukan pada metodologi pemrograman yang lain sebagai metodologi dasarnya yang membawa keuntungan sistem dasarnya. Sebagai contoh implementasi AOP bisa mengambil OOP sebagai sistem dasar untuk mendapatkan keuntungan implementasi *common concern* dengan OOP. Dengan implementasi tersebut, tiap *concern* menggunakan OOP untuk mengidentifikasi tiap *concern*. Ini analog dengan bahasa prosedural yang bertindak sebagai dasar bahasa untuk berbagai bahasa OOP.

2.4.5 Tujuan dan Sasaran Aspect Oriented Programming (AOP)

Tujuan dari AOP adalah menentukan semua kelas (dan hubungan serta tingkah laku yang berkaitan dengannya) yang relevan dengan masalah yang akan dipecahkan. Untuk itu perlu melakukan sejumlah tugas yaitu:

1. Persyaratan pemakaian dasar harus di komunikasikan diantara pelanggan dan perekayasa perangkat lunak.
2. Kelas–kelas harus diidentifikasi, misalnya atribut dan metode yang ditentukan.
3. Hirarki kelas harus dispesifikasikan.
4. Hubungan objek ke objek (koneksi objek) harus direpresentasikan.
5. Tingkah laku objek dimodelkan.

2.5 Metodologi Berorientasi Aspek

Metodologi Berorientasi Aspek merupakan metode pemograman yang menghadirkan konsep untuk memotong silang (*crosscut*) permasalahan *tangling* dan *scattering*.

2.5.1 Karakteristik Metodologi Berorientasi Aspek

Metodologi pengembangan sistem berorientasi aspek mempunyai tiga karakteristik utama yaitu :

(Coad-Yourdon,1991) :

a. Encapsulation

Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam satu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.

b. Inheritance

Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metoda dari induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metoda lagi pada anaknya, karena telah mewarisi sifat induknya. *Inheritance* mempunyai arti bahwa atribut dan operasi yang dimiliki bersama diantara kelas yang mempunyai hubungan secara hirarki. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan secara spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya.

Sifat yang dimilikinya oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya. Sifat yang dimilikinya oleh kelas induknya tidak perlu diulang dalam setiap subkelas. Sebagai contoh, Sedan dan Sepeda Motor adalah subkelas dari Kendaraan Bermotor.

Kedua subkelas mewarisi sifat yang dimiliki oleh Kendaraan Bermotor, yaitu:

1. Mempunyai mesin.
2. Dapat berjalan.

Kedua subkelas mempunyai sifat masing-masing yang berbeda, misalnya jumlah roda, dan kemampuan untuk berjalan mundur yang tidak dimiliki oleh sepeda motor. Beberapa faktor dari superkelas yang bersifat umum dan dimasukkan kedalam kelas induknya serta mewariskan sifat tersebut pada kelas turunannya, sehingga mengurangi pengulangan yang terjadi dalam desain dan pemrograman.

c. Polymorphism

*Polymorphism*s (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan prilaku yang berbeda. Polimorfisme mempunyai arti bahwa opersi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda. Operasi *move* mungkin mempunyai perbedaan dalam kelas windows atau kendaraan bermotor. Suatu implementasi yang spesifik dari suatu operasi dari kelas tertentu disebut metoda. Karena operator berorientasi objek adalah bersifat polimorfisme, mungkin dapat mempunyai lebih dari suatu metode.

2.6 *Development Aspect*

Development aspect digunakan selama proses *development* dan diharapkan untuk dihilangkan pada aplikasi yang final. Aspek ini dapat diaktifkan atau dinon-aktifkan pada tiap beda tahapan dari proses *development*. *Tracing*, *profiling*, *counting*, dan *logging* merupakan aspek yang dapat dijadikan contoh. Seringkali *programmer* menambahkan *trace statements* ke dalam kode program dan dihilangkan ketika *debugging* selesai. Mengganti *concern* ini dengan aspek dapat membantu pemisahan instruksi ekstra ini lebih baik. Banyak *programmer* menggunakan model 'Design by Contract' yang dipopulerkan oleh Bertand Meyer. Dalam model pemrograman ini *pre-* dan *post-conditions* pemanggilan suatu *method* diberikan secara eksplisit agar dapat berjalan sesuai dengan yang diharapkan. *Pre* dan *port-conditions* dapat diimplementasikan sebagai *aspect*.

Aspect dapat digunakan untuk memastikan *method* tertentu dipanggil dengan parameter yang benar dan memberikan atau mengembalikan kembalian (*return*) yang benar. Sebagai contoh, dalam suatu sistem yang berhubungan dengan pengukuran jarak, sangat penting untuk memastikan nilai masukan yang diberikan ke komponen perhitungan selalu bernilai positif guna menjaga sistem berjalan konsisten. Aspek semacam ini didapati sampai aplikasi yang final.

Mekanisme *crosscut* berbasis *property* dapat berguna dalam mendefinisikan pelaksanaan kontrak yang rumit. Satu kekuatan besar dari penggunaan mekanisme ini adalah untuk mengidentifikasi pengambilan dan penulisan property melalui pemanggilan method. Hal semacam ini akan menjaga konsistensi antara desain dan implementasi juga ketika *compile* dan *run-time*. Selain itu juga untuk menjaga konsistensi desain *layer* sebagai contoh, tidak diperkenankannya *Persentation Layer* langsung mengakses *Persistance Layer*, *aspect* dapat membantu untuk menjaga hal seperti ini ketika *compile-time*. Aspek semacam ini bisa dihilangkan ketika *run-time*.

2.6.1 *Production Aspect*

Production aspect lebih digunakan dalam konteks yang bersifat operasional, yaitu ketika user menjalankan aplikasi final. Java beans merupakan komponen perangkat lunak yang *reusable* yang secara visual dapat dimanipulasi oleh *tool* IDE. Sedikit *requirement* menjadikan obyek menjadi *sebuah* bean. *Bean* harus didefinisikan dengan konstruktor tanpa argument dan harus diturunkan dari Serializable atau Externalizable. *Property* yang ada di obyek diperlakukan sebagai *property bean* yang harus dindikasikan dengan adanya method *get* dan *set* yang penamaannya *getProperty* dan *setProperty* dimana property merupakan nama *field* dalam *class bean*. Beberapa *property* dari *bean*, yang dikenal dengan *bound property*, mengeluarkan *event* ketika terjadi perubahan nilai property, sehingga *listener* yang terdaftar akan dapat informasi perubahan tersebut. Dalam membuat *bound property*, melibatkan penyimpanan daftar *listener*, serta pembuatan dan *dispatch event* obyek di method yang nilai *property*-nya berubah. *Dispatch event* seperti ini biasa disebut sebagai *change monitoring*. *Change monitoring* yang

akan men-*crosscut* dekomposisi fungsional dapat digunakan untuk meningkatkan penanganan *persistence* dan *cache*.

Pertimbangan implementasi fungsionalitas yang memperbolehkan *client* dari suatu editor untuk merubah warna ke suatu *element* gambar biasanya ini membutuhkan *passing* suatu warna atau *color factory* dari *client* menuju ke pemanggilan suatu *method* di *element factory*. Semua programmer biasa menghadapi ketidaknyamanan dalam menambahkan satu argumen ke *method-method* yang berhubungan, hanya karena untuk melakukan *passing* informasi semacam ini. Dengan menggunakan *aspect*, *context passing* semacam ini dapat diimplementasikan dengan cara yang modular.

Property based aspect dapat digunakan untuk menjaga konsistensi penanganan fungsionalitas dari banyak operasi. Seperti untuk memastikan *log* setiap ada *exception* terjadi. Banyak *development* pada suatu aplikasi membutuhkan perubahan kecil pada *requirement* yang disesuaikan dengan kebutuhan permasalahan bisnis *client*. Hal ini cenderung sedikit merubah fungsionalitas pada satu atau dua *method*. AOP dapat menjawab isu ini daripada mengimplementasi ulang *class* atau menurunkan (*inherits*) dan meng-*override*-nya dari *class* induknya. *Aspect* dapat membantu *design pattern* yang menggunakan *class helper* yang membutuhkan banyak penambahan *code* untuk menangani kebutuhan *level* sistem. *Synhconous*, *distribution* (*load balancing*), *resource pooling*, *storage management*, *session management*, *performance*, *authentication*, dan administrasi merupakan contoh lain dari kebutuhan *level* sistem yang ada di *production* untuk aplikasi berskala *enterprise*. Setiap *aspect* men-*crosscut* berbagai sub-sistem, sebagai contoh, *storage management* mengenai setiap *business object* yang *stateful*.

2.6.2 Dasar AOP *(Aspect Orienied Programming)*

Para peniliti telah mempelajari berbagai macam cara untuk menyelesaikan masalah dibawah topic umum dari '*separation of concern*'. AOP merepresentasikan salah satu metode tersebut. AOP pada intinya, menyediakan implementasi *individual concern* dengan cara yang *loosely coupled*, dan mengkombinasikan implementasi ini untuk membentuk sistem yang final. AOP

membuat sistem menggunakan *loosely coupled* dan modularisasi implementasinya dari *cosscutting concern*. OOP sebaliknya, membuat sistem menggunakan *loosly coupled* dan modularisasi implementasinya dari *common concern*, *common concern* di OOP disebut kelas. Unit modularisasi dari AOP disebut sebagai aspek. Implementasi AOP melibatkan tiga tahap development:

1. *Aspectual decomposition*

   Memisahkan kebutuhan untuk mengidentifkasi *crosscutting* dan *common concern* (pemisahan *concern level* modul dari *crosscuting concern level* sistem). Sebagai contoh *credit card* modul, dapat didentifikasi menjadi tiga *concern*: inti pemrosesan *credit card*, *logging*, dan *authentication*.

2. *Concern implementation*

   Implementasi tiap *concern* secara terpisah. Untuk contoh *credit card*, diimplementasikan unit inti pemrosesan *credit card*, unit *logging* dan unit *authentication*.

3. *Aspectual re-composition*

   Pada tahap ini *aspect integrator* menspesifikasikan aturan rekomposisi dengan membuat unit modularisasi yaitu aspek. Proses rekomposisi juga dikenal sebagai *weaving*, menggunakan informasi ini untuk membentuk sistem yang final.

   (Mahendra, 2004)

2.6.3 Anatomi Bahasa AOP

Seperti halnya metodologi pemrograman yang lainnya, implementasi AOP terdiri dari dua bagian: spesifikasi bahasa dan implementasi. Spesifikasi bahasa menjelaskan bagaimana suatu bahasa dibangun dan juga *syntax*-nya. Implementasi memverifikasi kode program berdasarkan spesifikasinya dan merubahnya menjadi bentuk yang dapat dieksekusi di mesin tujuan, umumnya dilakukan oleh *compiler* atau *interpreter*.

Pada level yang lebih tinggi, suatu bahasa AOP menspesifikasikan dua komponen:

1. *Implementation of concerns*

   Pemetaan masing-masing kebutuhan manjadi kode program sehingga *compiler* dapat merubahnya menjadi kode yang dapat dieksekusi oleh mesin. Karena implementasi *concern* mengambil bentuk dari prosedur, maka bahasa tradisional seperti C, C++, atau Java dapat digunakan dengan AOP.

2. *Weaving rules specification*

   Bagaimana menyusun secara independen *concern-concern* yang telah diimplementasikan ke bentuk sistem yang final. Untuk mencapai tujuan ini, implementasinya dapat menggunakan atau membuat bahasa untuk menentukan aturan-aturan penyusunan pecahan implementasi yang berbeda tersebut ke dalam bentuk sistem yang final. Bahasa yang menentukan aturan *weaving* bisa hanya merupakan ekstensi dari bahasa yang ada atau bisa berupa bahasa yang baru atau berbeda sama sekali.

   Compiler bahasa AOP melakukan dua tahapan logika:

1. Menggabungkan masing-masing *concern*
2. Merubah informasi hasilnya menjadi kode yang dapat dieksekusi.

Suatu implementasi AOP dapat mengimplementasikan *weaver* dalam berbagai cara, termasuk translasi dari *source code* ke *source code*. Di sini pertama kita menyiapkan *source code* untuk tiap aspek untuk menghasilkan *source code* yang di-*weaving*. Kemudian *compiler* AOP menaruh kode hasil konversi ini ke *compiler* bahasa dasarnya untuk menghasilkan kode final yang dapat dieksekusi.
Sebagai contoh yang menggunakan pendekatan ini, suatu implementasi AOP berbasis Java akan menkonversi tiap-tiap *source code aspect* ke *source code* Java, kemudian Java *compiler* merubahnnya menjadi *byte-code*. Pendekatan yang sama juga dapat dilakukan untuk *level byte-code*, bagaimanapun *byte-code* juga merupakan bentuk lain dari *source code*. Selain itu, sistem eksekusi katakanlah VM dapat dibuat untuk mengenali *aspect*. Dengan pendekatan ini, untuk implementasi AOP berbasis Java misalnya, VM pertama-tama akan mengambil aturan-aturan *weaving*, dan memberlakukannya untuk setiap pemanggilan *class-class* berikutnya. Dengan kata lain, ini merupakan *just-in-time weaving*.

2.6.4 Kelebihan dan Kekurangan *Metodologi Berorientasi* Aspek

2.6.4.1 Kelebihan

1. Tanggung jawab yang lebih bersih dari tiap modul
2. Modularisasi yang lebih baik
3. Mudahnya sistem untuk berevolusi
4. Kode lebih reusable yang artinya kemampuan untuk menggunakan ulang hasil analisis, perancangan, serta pemrograman (*reusable component*) pada suatu proyek ke proyek lainnya.

(Ridwan, 2006)

Dengan menggunakan berorientasi aspek maka dalam melakukan pemecahan suatu masalah dalam Sistem Informasi Perawtan Pasien RSIA Labuh Baru yaitu dengan melihat bagaimana cara menyelesaikan suatu masalah tersebut dengan terstruktur tanpa memisahkan objek-objek dan kelas yang berhubungan satu sama lain yang dapat melakukan memecahan masalah tersebut dan untuk membuat suatu program yang terdiri dari berbagai object yang saling berinteraksi dengan bertukar pesan antar object.

2.6.4.2 Kekurangan

1. Alur program di AOP sulit untuk ditelusuri tidak seperti di OOP atau bahkan di procedural. Sebenarnya pada OOP juga terjadi hal demikian, polymorphic method membuat analisa alur program menjadi rumit, begitu juga dengan bahasa prosedural seperti C, adanya function pointer, *flow* program menjadi tidak statis dan sulit dipahami.
2. AOP tidak menjawab semua masalah baru itu benar, ia hanya menjawab permasalahan yang sebelumnya tidak terpecahkan dengan cara yang lebih baik dengan lebih sedikit usaha dan meningkatkan pemeliharaanya. Kita bisa menyelesaikan semua permasalahan dengan metodologi apapun, namun yang membedakan adalah kompleksitas solusinya.

   (Ridwan, 2006)

2.7 Model Umum Perancangan Analisis dan Perancangan Sistem

Adapun model perancangan analisis dan perancangan sistem ini adalah:

2.7.1 *Rational* Rose

Rational Rose merupakan salah satu *tool* yang digunakan membangun model suatu sistem secara visual yang memiliki banyak kemampuan untuk pembentukan sistem berorientasi obyek yang menggunakan *UML*. Dalam *UML* terdapat beberapa istilah yang sering digunakan seperti : *views*, *diagram* dan *elemen model*.

1. *View*

   Rational Rose memiliki empat *view* yaitu : *Use Case View, Logical View, Componen View* dan *Deployment View.*

2. *Diagram Rational Rose* memiliki delapan diagram yaitu : *Use case diagram, Sequence diagram, Collaboration diagram, Activity diagram, Class diagram, State diagram, Component diagram* dan *Deployment diagram*.

3. *Elemen Model*

   Konsep-konsep yang digunakan dalam diagram merupakan elemen-elemen model yang menyatakan konsep berorientasi obyek secara umum, seperti *class*, *object* dan *message*, serta hubungan antar konsep termasuk *association*, *dependency* dan *generalization*.

2.7.2 Pemodelan Metodologi Berorientasi Aspek

Pemodelan adalah suatu metode untuk menggambarkan struktur sistem yang memperlihatkan hubungan objek terhadap objek yang lain, serta menampilkan atribut dan operasi yang menjadi ciri suatu kelas tertentu untuk kepentingan pengembangan suatu sistem informasi atau perangkat lunak. Pemodelan objek menggambarkan secara abstraktif tentang fakta-fakta yang ada pada dunia nyata sehingga mudah diimplementasikan dalam bentuk perangakat lunak atau sistem informasi.

2.7.2.1 UML (*Unified Modelling Language)*

UML adalah bahasa untuk memvisualisasi, menspesifikasikan, dan mendokumentasi artifak-artifak sistem perangkat lunak. *UML* merupakan sistem notasi yang membantu pemodelan sistem menggunakan konsep berorientasi objek.

UML (*Unified Modelling Language*) merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan visualisasi, mengkonstruksi, dan mendokumentasikan *artifact* yang terdapat dalam sistem software. UML merupakan pemodelan yang paling sukses dari tiga metode object oriented yang telah ada sebelumnya, yaitu Booch, OMT, dan OOSE. UML merupakan kesatuan dari ketiga metode pemodelan tersebut dan ditambah kemampuan lebih karena mengandung metode tambahan untuk mengatasi maslah pemodelan yang tidak dapat ditangnai ketiga metode tersebut.

Artifact adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software, *artifact* dapat berupa model, deskripsi, atau software. (Sutopo, 2002).

UML merupakan pengembangan dari metode-metode peranvangan yang sebelumnya seperti metode Booch, Rumbaugh atau OMT (*Object Modelling Technique),* OOSE yang berkembang tahun 90-an. Standar UML memberikan ketentuan pada semi formal semantic meta model yang memberikan konsep dasar pemodelan seperti *class* dan *object* serta konsep pemodelan dengan notasi grafis yang terdiri dari 8 tipe diagram.

UML terdiri atas dari beberapa diagram, diantaranya yaitu (Nugroho,2004):

1. *Use Case Views*

Model ini berfungsi untuk menggambarkan *system design outside user* (pemakai luar) yang disebut *actor.* Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai dengan yang diinginkan *external actors.* Aktor

yang yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digunakan terutama untuk pelanggan, perancang (*designer)*, pengembang *(developer),* dan penguji sistem (*tester).*

Diagaram *usecase* digunakan untuk menggambarkan hubungan transaksi antar sistem dan *end user*, selain itu diagram *usecase* dapat diartikan sebagai ganbaran *actor* dengan kumpulan *usecase* yang menyertakan batasan sistem, kumpulan komunikasi antar *actor* dan *usecase* dan generalisasi dalam *usecase* (Suhendar, 2002).

Simbol-simbol yang terdapat dalam diagram *usecase* yaitu :

a. *Usecase*

Usecase adalah spesifikasi rangakain dari tindakan baik termasuk rangkaian berbeda, subsistem atau *class* dapat lakukan dari hubungan dengan pihak luar. *Usecase* adalah proses-proses yang terjadi dalam suatu software. *Usecase* juga menggambarkan apa yang sedang dilakukan oleh seorang aktor.

Usecase merupakan sekumpulan skenario yang dihubungkan satu sama lain dengan satu tujuan yang sama dari pengguna. Penggunaan utama dari diagram *usecase* adalah untuk menyediakan sarana dalam mendokumentasikan dan memahami persyaratan sistem informasi yang sedang berkembang. *Usecase* dan diagram *usecase* adalah beberapa alat yang paling penting untuk digunakan dalam analisis dan desain sistem berorientasi objek dan aspek.

Cara menentukan *usecase* dalam suatu sistem:

1. Pola prilaku perangkat lunak aplikasi.
2. Gambaran tugas dari sebuah actor.
3. Sistem atau benda yang memberikan sesuatu yang bernilai kepada aktor.
4. Apa yang dilakukan oleh suatu perangkat lunak (bukan bagaimana cara mengerjakannya).

b. Actor / Aktor

Aktor merupakan bagian dari *usecase* yang bertindak sebagai subjek (pelaku) dalam suatu proses.

Aktor adalah abstraksi untuk *entity* luar dari sistem, subsistem atau *class* yang berhubungan secara langsung dengan sistem. Pada *usecase* diagram diperlukan beberapa aktor dimana aktor tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah aktor mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima dan memberi informasi pada sistem, aktor hanya berintereaksi dengan *usecase* tetapi tidak memiliki kontrol atas *usecase.* Aktor digambarkan dengan *stick man*. Aktor dapat digambarkan secara umum atau spesifik, dimana untuk membedakannya dapat menggunakan relationship.

Ada beberapa kemungkinan yang menyebabkan *actor* tersebut terkait dengan sistem antara lain :

1. Yang berkepentingan terhadap sistem dimana adanya arus informasi baik yang diterimanya maupun yang diinputkan kesistem.
2. Orang ataupun pihak yang mengelola sistem tersebut.
3. *External resource* yang digunakan oleh sistem.
4. Sistem lain yang berinteraksi dengan sistem yang akan dibuat.

2. Static Views

Static Views adalah gambaran tentang keseluruhan model yang digolongkan berdasarkan sistem dan hubungan statis mereka. Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,* dan *relationship)* dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class* diagram untuk struktur statis dan dalam *state, sequence, collaboration,* dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer)* dan pengembang (*developer)*. Unsur pokok dari *static view* adalah *classes* dan *relationship. Static view* digambarkan derngan menggunakan *class diagram,* karena *class diagram* adalah deskripsi dari *classes.*

Class diagram adalah diagram yang memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain suatu sistem. *Relationship* adalah hubungan semantik antara model elemen.

3. *Interaction Views*

   Interaction Views mendeskripsikan tentang rangkaian dari pertukaran perintah berdasarkan peraturan yang merupakan jalan dari pelaksanaan sistem. *Interaction views* ini digambarkan dengan menggunakn *sequence diagram. Sequence diagram* itu sendiri dapat diartikan sebagai diagram yang menjelaskan interaksi objek yang disusun dalam suatu urutan waktu (Bahrami,1999).

4. *State Machine Views*

   State Machine Views adalah model tentang sejarah dari *object* dan *class*. *State machine mengandung state* yang dihubungkan dengan transisi. *State machine views* digambarkan dengan *state chart diagram.*

   State chat diagram adalah diagram yang memperlihatkan urutan keadaan sesaat (*state)* yang dilalui sebuah objek, kemudian yang menyebabkan sebuah ttransisi dari suatu *state* atau aktivitas kepada yang lainnya dan aksi yang menyebabkan sebuah transisi dari suatu *state* atau aktivitas kepada yang lainnya dan aksi yang menyebabkan perubahan satu *state* atau aktivitas.

5. *Activity Views*

   Activity Graph berbeda dengan *state machine* yang menggambarkan perhitungan aktivitas termasuk didalam pelaksanaan perhitungan. *Activity state* mewakili aktivitas langkah aliran kerja atau eksekusi dari sebuah operasi. *Activity view* digambarkan dengan *activity diagram. Activity diagram* itu sendiri memiliki arti sebagai diagram yang memodelkan alur kerja (*work flow)* sebuah proses bisnis dan urutan aktifitas dalam satu proses.

6. *Deployment View*

   Deployment view memperlihatkan pemetaan setiap proses kedalam *hardware. View* ini paling bermanfaat ketika membuat model suatu sistem

yang diterapkan dalam lingkunagn arsitektural yang terdistribusi yaitu ketika menerapkan aplikasi dan server pada lokasi yang berbeda. *View* ini hanya memiliki satu diagram, yaitu *deployment diagram.*

Dalam proyek pengembangan sistem apapun, fokus utama dalam analisis dan perancangan adalah model. Hal ini berlaku umum tidak hanya untuk perangkat lunak. Dengan model kita dapat merepresentasikan sesuatu karena :

a. Model mudah dan cepat untuk dibuat.
b. Model bisa digunakan sebagai simulasi untuk mempelajari lebih detail tentang sesuatu.
c. Model bisa dikembangkan sejalan dengan pemahaman kita tentang sesuatu.
d. Model bisa mewakili sesuatu yang nyata maupun tidak nyata.

Konsep-konsep yang diterapkan di *UML* adalah satu model berisikan informasi mengenai sistem (atau domain), model-model berisi elemen-elemen model seperti kelas, simpul-simpul, paket-paket, dan sebagainya. Satu diagram menunjukkan satu pandangan tertentu dari model.

Dalam membangun suatu model perangkat lunak dengan *UML*, digunakan bentuk-bentuk diagram atau simbol untuk merepresentasikan elemen-elemen dalam sistem.

Bentuk diagram yang digunakan untuk merepresentasikannya adalah sebagai berikut :

1. *Use case Diagram*
2. *Activity Diagram*
3. *Sequence diagram*
4. *Class Diagram*
5. *Collaboration Diagram*
6. *State Diagram*
7. *Component Diagram*
8. *Deployment Diagram*

Tipe diagram UML:

Tabel 2.1 Tipe Diagram *UML*

Diagram	**Tujuan**
Use Case	Menunjukkan sekumpulan kasus fungsional dan aktor dan hubungannya.
Activity	Pandangan operasi, bagaimana objek-objek bekerja, aksi-aksi yang mempengaruhi obyek, pandangan *use case workflow*.
Sequence	Berfungsi untuk *overview* prilaku sistem, menunjukkan objek-objek yang diperlukan, mendokumentasikan skenario dari suatu diagram *Use Case*, memeriksa jalur-jalur pengaksesan.
Class	Memodelkan kosakata di sistem, distribusi dan tanggung jawab, tipe primitif, kolaborasi, skema *database* logik.
Collaboration	Memodelkan pandangan prilaku sistem pada *link-link* di antara objek-objek. Ilustrasi dari *use case*, memeriksa jalur-jalur pengaksesan
Statechart	Pandangan objek secara waktu, pandangan dalam berkaitan dengan ransangan eksternal.
Component	Memodelkan *file* yang dapat dieksekusi dan pustaka, memodelkan tabel, *file* dan dokumen, memodelkan API (*Application Programming Interupt*)
Deployment	Konfigurasi pemprosesan saat jalan dan komponen-komponen yang terdapat didalamnya.

Sumber: Adi Nogroho (2005)

2.8 Model Basis Data Relasional

Model basis data relasional (*Relational Data Base Model / RDBM*) menunukkan / suatu cara / mekanisme yang digunakan untuk mengelola / mengorganisasi data secara fisik dalam memori sekunder yang akan berdampak pula pada bagaimana kita mengelompokkan dan membentuk secara keseluruhan data yang terkait dalam system yang sedang ditinjau.

Ada berbagai model basis data diantaranya model basis data hirarki dan model basis data jaringan (*network*) serta model basis data RDBM. Namun model basis data yang paling banyak digunakan / diterapkan adalah model basis data RDBM, karena konsep dan terminologi yang digunakan hampir sama dengan kondisi yang sesungguhnya yang dihadapi oleh para pemakai, sehingga memudahkan para pemakai dalam memahaminya.

RDBM menjelaskan kepada pemakai tentang hubungan logik antar data dalam basis data dengan mempresentasikannya ke dalam bentuk relasi-relasi

berupa tabel mendatar (*flat file*) yang terdiri atas sejumlah baris yang menunjukkan atribut tertentu. Relasi dirancang sedemikian rupa sehingga tidak terjadi kerangkapan data. Dalam sebuah baris data, kerelasian antar relasi satu dengan yang lainnya ditunjukkan menggunakan *Foreign Key / F*K atau relasi bertipe transaksi.

2.9 *Hypertext Prepprocessor* (PHP)

Script PHP menunjukkan bahwa bahasa web *serverside* yang bersifat *open source*. Bahasa PHP menyatu dengan *script* HTML yang sepenuhnya dijalankan pada server.

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama FI (*Form Interpreted*), yang terwujudnya berupa sekumpulan *script* yang digunakan untuk mengolah data *form* dari web. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP / FI, kependekan dari *Personal Home Page / Form Interpreter*. Dengan perilisan kode sumber ini menjadi *open source*, maka banyak programmer yang tertarik untuk ikut mengembangkan PHP.

Pada November 1997, dirilis PHP / FI 2.0. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP / FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang palin banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model

pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman kea rah paradigm berorientasi objek.

2.9.1 Karakter

Istilah karakter banyak dijumpai pada bahasa pemrograman komputer. Yang disebut karakter itu dapat berupa sebuah huruf, sebuah angka tunggal, sebuah spasi, tanda *control* seperti *carriage return* atau sebuah simbol seperti + dan ?.

2.9.2 Pengenal

Pengenal (*identifier*) banyak digunakan dalam program untuk member nama variable, funsi atau kelas. Oleh karena itu mengetahui aturan penamaan pengenal adalah hal yang penting. Aturan yang berlaku untuk pengenalan yaitu :

a. Karakter yang dapat digunakan adalah huruf, angka atau garis bawah.
b. Karakter pertama harus berupa huruf atau garis bawah.
c. Panjang pengenal bisa berapa saja.
d. Huruf kecil dan huruf kapital dibedakan.

Bila membuat suatu nama, hindarkan penamaan nama yang sama dengan nama-nama yang sudah tersedia pada PHP. Misalnya, jangan menggunakan nama variabel berupa print, karena akan membingungkan pembaca program.

2.9.3 Tipe Data

Ketetapan memilih tipe data akan sangat menentukan pemakaian *resource* (sumber daya) oleh aplikasi yang dibuat sehingga akan dihasilkan program aplikasi yang efisien dan berperformance tinggi. PHP memiliki delapan tipe data, yaitu :

1. Integer

   Integer memilki tipe data bilangan bulat dengan jangkauan kira-kira dari -2 milyar hingga +2 milyar.

2. Double

   Double menyatakan tipe data bilangan real atau titik mengambang, yaitu bilangan yang mempunyai bagian pecahan.

3. String

String menyatakan tipe data teks (sederetan karakter yang tidak menyatakan bilangan).

4. Boolean
5. Object
6. Array
7. Null
8. Resource

2.9.4 Konstanta

Konstanta menyatakan nilai yang tetap di dalam program. Pada PHP dikenal sejumlah karakter yang menggunakan penulisan secara khusus, yaitu didahului dengan simbol *blackslash* (\). Berikut ini adalah tabel yang berisi daftar karakter khusus dan cara penulisannya :

Tabel 2.2 Karakter-karakter yang diawali dengan tanda

Escape Sequence	Keterangan
\ddd	Karakter Oktal (ddd)
\uxxxx	Karakter Unicode Heksadesimal (dddd)
\’	Petik tunggal
\”	Petik ganda
\\	Backslash
\r	Carriege return
\n	Baris baru disebut juga dengan new line/line feed seperti penggunaan enter pada keyboard
\f	Form feed
\t	Tab
\b	Backspace

2.9.5 Variabel

Variabel digunakan dalam program untuk menyimpan nilai yang berubah-ubah. Variabel dapat dibayangkan seperti suatu kontak. Mula-mula kita menyimpan nilai 10 ke dalam kotak tersebut. Di saat yang lain kita mengganti isinya dengan 33.

2.9.6 Operator

Operator adalah simbol yang digunakan dalam program untuk melakukan suatu operasi, misalnya penjumlahan atau perkalian, perbandingan kesamaan dua buah nilai atau bahkan memberikan nilai ke variabel.

Macam-macam operator

1. Operator Penugasan : Untuk memberikan nilai ke suatu variabel.

Tabel 2.3 Daftar operator penugasan

Operator	Kegunaan	Contoh
=	Memberikan nilai ke suatu variabel	X = 2 ;
+=	Memberikan variabel di sisi kiri dengan nilai di sisi kanan	X += 2 ; Identik dengan : X:=X+2 ;
-=	Mengurangi isi variabel di sisi kiri dengan nilai di sisi kanan	X -= 2; Identik dengan : X:=X – 2
/=	Membagi variabel di sisi kiri dengan nilai di sisi kanan	X / = 2 ; Identik dengan : X:= X / 2 ;
%=	Memperoleh sisa pembagianantara variabel di sisi kiri dengan nilai di sisi kanan	X %= 2 ; Identik dengan : X:= X % 2 ;
&=	Melakukan operasi “dan” / ”and” terhadap variabel Di sisi kiri dengan nilai di sisi kanan	X &=2 ; Identik dengan : X:= & 2 ;
|=	Melakukan operasi “atau” / ”or” terhadap variabel Di sisi kiri dengan nilai di sisi kanan	X |=2 ; Identik dengan : X:= | 2 ;
^=	Melakukan operasi “atau ekslusif” / ”xor” terhadap variabel di sisi kiri dengan nilai di sisi kanan	X ^=2 ; Identik dengan : X:= ^ 2 ;
.=	Melakukan operasi konkatenasi terhadap variabel di sisi kiri dengan nilai di sisi kanan	X .=’A’ ; Identik dengan : X:= X.’A’ ;

2. Operator Aritmatika :Operator yang digunakan dalam operasi matematika.

Tabel 2.4 Operator-operator aritmatika

Operator	Keterangan	Contoh
*	Perkalian	7*9
/	Pembagian	8/2
+	Penjumlahan	6+3
-	Pengurangan	10-3
%	Modulus (sisa hasil bagi)	7/2

3. Operator Perbandingan : Membandingkan suatu data dengan data yang lain dan menghasilkan nilai logika (*boolean*) benar / salah

Tabel 2.5 Daftar operator perbandingan

Operator	Arti	Contoh

=	Sama dengan	Total = 50 (Nilai Total = 50)
>	Lebih besar dari	Total > 50 (Nilai total diaras 50, Misalnya 51, 52,….)
<	Lebih kecil dari	Total <50 (Nilai total dibawah 50, Misalnya 49, 10, 2,….)
>=	Lebih besar atau sama	Total <= 50 (Nilai total sama atau dibawah 50, Misalnya 0, 49, 34,….)
<>	Tidak sama dengan	Total < > 50 (Nilai total tidak sama dengan 50, Misalnya 7, 51, 83,….)

4. Operator Logika : Digunakan untuk menggunakan kondisi berganda dan menghasilkan sebuah ekspresi yang bernilai benar (nilai 1) atau salah (nilai 0).

Tabel 2.6 Operator-operator logika

Operator	Keterangan
AND atau &&	Menghasilkan nilai benar jika kedua operand bernilai benar
OR atau ||	Menghasilkan nilai benar kalau ada operand bernilai benar
XOR atau ^	Menghasilkan nilai benar jika hanya salah satu diantara operand bernilai benar
NOT atau !	Dikenakan di depan sebuah operand Misalnya : !$h Menghasilkan jika salah $h bernilai benar dan sebaliknya

2.10 *MySQL*

MySQL adalah salah satu jenis database server yang sangat terkenal. Kepopulerannya disebabkan karena *MySQL* menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya. Selain itu ia bersifat *free* (tidak perlu membayarnya untuk menggunakannya).

MySQL termasuk jenis RDBMS (*Relational Database Management System*). Itulah sebabnya istilah seperti tabel, baris dan kolom digunakan pada *MySQL*.

2.10.1 Tabel

Tabel adalah sebuah objek yang berfungsi untuk mendefinisikan dan menyimpan data menurut aturan tertentu.

2.10.2 Query

Query berfungsi untuk menyajikan data yang berasal dari satu atau lebih tabel sesuai dengan yang diinginkan. Query bisa berfungsi untuk memilih data, menghapus dan menyortir.

BAB III
METODOLOGI PENELITIAN

Metodologi penelitian adalah cara yang digunakan dalam memperoleh berbagai data untuk diproses menjadi informasi yang lebih akurat sesuai permasalahan yang akan diteliti. Metodologi penelitian dengan mendeskripsikan masalah yang dilengkapi dengan penyajian diagram alur pelaksanaan penelitian untuk memudahkan dalam memahami tahapan penelitian.

Penelitian ini juga menggunakan jenis penelitian deskriptif, yaitu jenis penelitian yang menuturkan pemecahan masalah yang ada sekarang berdasarkan data-data, menganalisis dan menginterpretasikannya yang bertujuan untuk memecahkan masalah secara sistematis dan faktual mengenai fakta-fakta dan sifat-sifat populasi.

Jenis Data dan Sumber Data

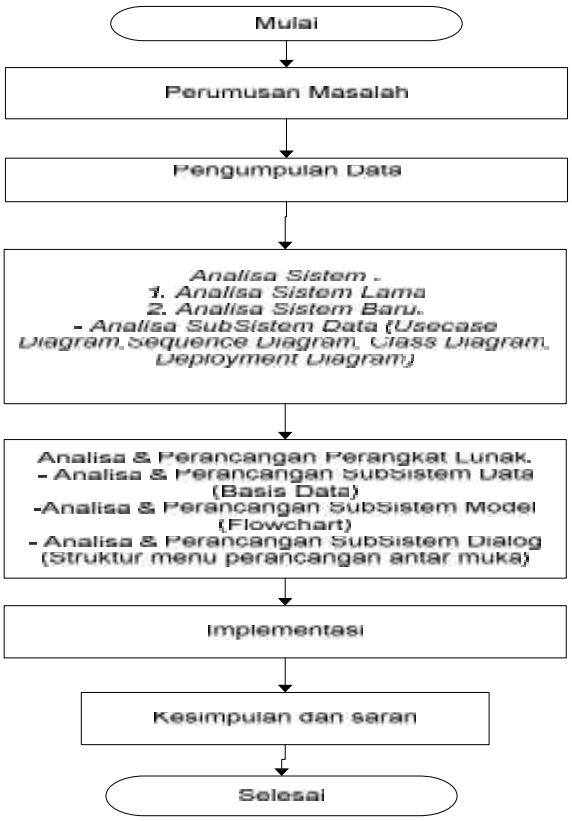
Dalam penelitian ini dibutuhkan beberapa jenis data dan sumber data, antara lain sebagai berikut :

1. Data Primer

   Data primer adalah data yang diperoleh secara langsung dari objek penelitian, seperti melakukan observasi ke Rumah Sakit Ibu dan Anak Labuh Baru, wawancara dengan Direktur RSIA Labuh Baru dan *Supervisor* RSIA Labuh Baru. Dalam hal ini data primer diperoleh langsung dari Rumah sakit Ibu dan Anak Labuh Baru.

2. Data Sekunder

   Data sekunder adalah data yang diperoleh secara tidak langsung, dalam hal ini terlebih dahulu data dikumpulkan dan dilaporkan yang didapat dari jurnal, internet maupun literatur yang masih berhubungan erat dengan tema proposal. Contohnya data pasien, data pegawai, data kamar, data fasilitas, data layanan, didapat dari literatur-literatur yang berhubungan dengan perawatan pasien rumah sakit.



Gambar 3.1 *Flowchart Metodologi* Penelitian

3.1 Perumusan Masalah

Merumuskan masalah tentang bagaimana menganalisa dan merancang sistem informasi perawatan pasien dengan menggunakan pendekatan aspek.

3.2 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data tentang analisa dan perancangan sistem perawatan pasien rumah sakit ibu dan anak. Semua tahap pada proses pengumpulan data tersebut diperoleh dari wawancara dan studi pustaka.

a. Wawancara (*interview*)

Proses wawancara dilakukan kepada Direktur RSIA dan Supervisor RSIA Labuh Baru tentang sistem informasi perawatan pasien dengan menggunakan pendekatan berorientasi aspek.

b. Observasi (*Observation*)

Observasi yang dilakukan adalah mengumpul data tentang data pasien, data pegawai, data kamar, data pelayanan, data fasilitas, dan laporan.

c. Studi Pustaka (*Library Research*)

Studi pustaka dilakukan dengan tujuan untuk mengetahui metode apa yang akan digunakan dalam menyelesaikan permasalahan yang akan diteliti, serta mendapatkan dasar-dasar referensi yang kuat dalam menerapkan suatu metode yang akan digunakan dalam tugas akhir ini, yaitu dengan mempelajari buku-buku, artikel-artikel, dan jurnal-jurnal yang berhubungan dengan permasalahan yang akan dibahas.

3.3 Analisa Sistem

Analisa dan perancangna sistem dalam tugas akhir ini terbagi dua, yaitu analisa sistem lama dan analisa sistem baru.

3.3.1 Analisa Sistem Lama

Analisa sistem lama adalah menganalisa sistem yang sedang diterapkan di RSIA Labuh Baru yaitu melakukan pendaftaran yang masih dilakukan secara manual yaitu masih disimpan dalam bentuk kertas folio yang dimasukkan kedalam map pasien. Tentu dalam hal ini akan membutuhkan waktu yang lama untuk melakukan pencatatan data pasien.

3.3.2 Analisa Sistem Baru

Analisa sistem baru adalah menganalisa sistem yang akan dibangun dengan menerapkan metode berorientasi aspek. Adapun analisa sistem baru yang akan digunakan meliputi:

1. Analisa subsistem data

Pada tahap ini dilakukan analisa terhadap data-data yang diperlukan agar sistem dapat berjalan sesuai harapan yang dimodelkan. Perancangan yang digunakan yaitu dengan tool UML yang dilakukan dalam bentuk pembuatan diagram. Setelah itu dilanjutkan merancang usecase diagram, sequence diagram dan class diagram.

1. *Use case Diagram*

   Use case Diagram merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use-case*, aktor, dan hubungannya.

2. *Sequence Diagram*

   Sequence diagram mendokumentasikan komunikasi/interaksi antar kelas-kelas.

3. *Class Diagram*

   Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah obyek dan merupakan inti dari pengembangan dan desain berorientasi obyek.

4. *Deployment Diagram*

   Konfigurasi pemprosesan saat jalan dan komponen-komponen yang terdapat didalamnya.

3.4 Perancangan Perangkat Lunak

Tahap analisa dan perancangan sistem informasi perawatan pasien merupakan tahapan dalam membuat rincian Sistem Informasi dari ketiga subsistem (basis data, model, dan komunikasi atau dialog) agar dimengerti oleh pengguna (*user*).

1. Tahapan analisa dan rancangan dari subsistem data adalah merancang basis data yang akan digunakan.
2. Tahapan subsistem model adalah merancang *fllowchart* sistem dengan menerapkan model objek *oriented*.
3. Tahapan subsistem dialog adalah merancang tampilan antar muka sistem (*user interface*) dan struktur menu.

3.5 Implementasi

Pada proses implementasi ini akan dilakukan pembuatan modul-modul yang telah dirancang dalam tahap perancangan ke dalam bahasa pemrograman.

3.6 Kesimpulan dan Saran

Dalam tahap ini dapat ditentukan kesimpulan terhadap hasil pengujian yang telah dilakukan, untuk mengetahui apakah implementasi sistem yang telah

dilakukan dapat beroperasi dengan baik serta memberikan saran-saran untuk menyempurnakan dan mengembangkan penelitian selanjutnya.

Rencana Kerja

Rencana kerja laporan penelitian tugas akhir ini diperlukan penulis dalam melaporkan hasil-hasil tugas akhir yang dilakukan dalam membangun Sistem Informasi Perawatan Pasien RSIA(Rumah Sakit Ibu dan Anak) Labuh Baru dengan menggunakan pendekatan berorientasi aspek.

Tabel 3.1 Rencana Kerja

No	Tahapan	Nov	Des	Jan	Feb	Mar	Apr	Mei	Jun	Juli	Agus	Sept	Okt
1.	Pengajuan Judul	√											
2.	Pengumpulan Data	√	√	√									
3.	Seminar Proposal				√								
4	Analisa dan Perancangan				√	√	√	√	√				
5.	Implementasi dan Pengujian							√	√	√			
6.	Laporan				√	√	√	√	√	√	√	√	
7	Seminar Hasil dan Sidang												√

BAB IV

ANALISA DAN PERANCANGAN

4.1 Analisa Sistem

Analisa dan perancangan yang dikembangkan adalah sebuah analisa dan perancangan sistem informasi perawatan pasien pada RSIA Labuh Baru yang dapat dimanfaatkan untuk melakukan pengolahan data perawatan pasien yang ada di RSIA Labuh Baru. Perawatan pasien yang dimaksud adalah perawatan pasien rawat inap dan perawatan pasien rawat jalan.

Analisa sistem dilakukan oleh analis untuk menentukan proses yang harus dikerjakan untuk memecahkan permasalahan-permasalahaan yang ada. Sasaran yang dilakukan setelah tahap analisis sistem adalah untuk meyakinkan bahwa analisis sistem telah berjalan pada jalur yang benar.

Seperti yang telah dijelaskan dalam bab satu, bahwa proses pencatatan data perawatan pasien rumah sakit ibu dan anak di Labuh Baru masih dilakukan dengan cara manual mulai dari pencatatan perawatan data pasien, data harga dan laporan data pasien maupun pegawai masih dicatat dalam suatu kertas folio dan disimpan dalam map pasien yang dilakukan oleh petugas administrasi. Hal ini menyebabkan tidak optimalnya pelayanan terhadap semua pihak yang mempunyai keterkaitan dengan RSIA tersebut, baik masyarakat yang berobat maupun instansi yang berkepentingan.

Pengolahan data secara manual dinilai tidak efisien lagi, maksudnya adalah proes pencarian data yang dibutuhkan untuk proses pengolahan data pegawai dan pasien membutuhkan waktu yang lama karena data masih disimpan dalam bentuk arsip. Di samping itu akan menimbulkan ketidakefektifan dalam penyimpanan data yang dilakukan. Selain karena kertas yang bersifat lapuk dan mudah rusak penghematan dari segi materi dan waktu juga kurang efektif. Sehingga sangat

diperlukan perangkat lunak yang dapat mengolah data-data rumah sakit tersebut secara optimal khususnya data perawatan pasien.

Berikut akan diuraikan secara umum gambaran prosedur yang berkaitan dengan analisa dan perancangan sistem perawatan pasien rumah sakit ibu dan anak, secara umum meliputi proses-proses sebagai berikut :

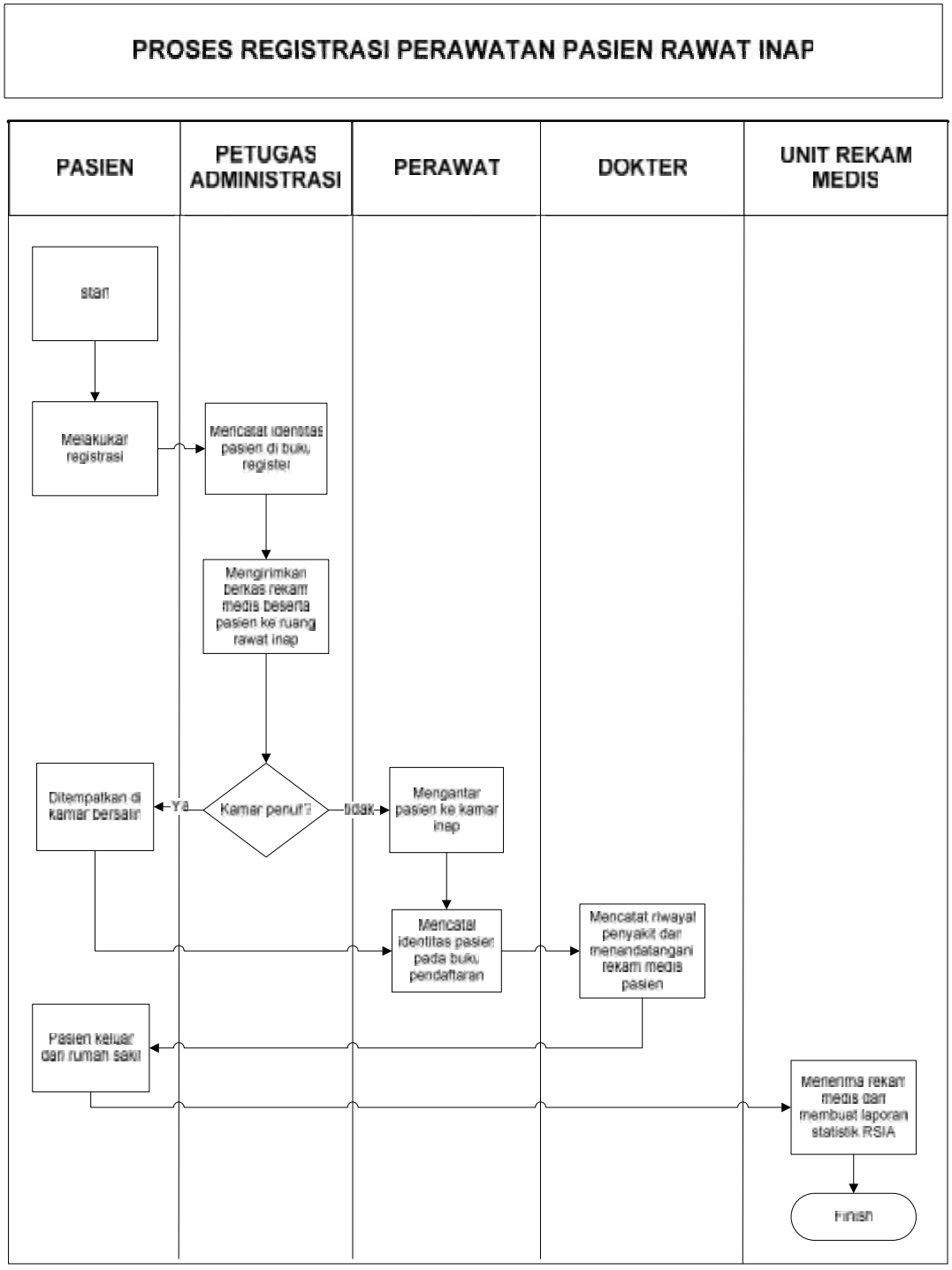
1. Proses registrasi perawatan pasien rawat inap.

   Pada proses registrasi perawatan pasien rawat inap ini, langkah-langkah yang dilalui oleh pasien adalah sebagai berikut :

   a. Petugas bagian administrasi menerima pasien dan mencatat dalam buku register penerimaan pasien rawat inap : Nama, Nomor Rekam Medis, Identitas pasien pada lembaran masuk. Petugas penerimaan pasien rawat inap mengirimkan berkas rekam medis, bersama-sama dengan pasiennya ke ruang rawat inap yang dimaksud.
   b. Pasien membawa surat permintaan rawat inap dari dokter poliklinik, Unit Gawat Darurat menyediakan tempat penerimaan pasien rawat inap.
   c. Pasien diterima oleh perawat di ruang rawat inap dan dicatat pada buku pendaftaran. Jika kamar penuh pasien sementara ditempatkan dikamar bersalin. Pada proses ini, dokter yang bertugas mencatat tentang riwayat penyakit pasien, hasil pemerikasaan fisik serta semua tindakan yang diberikan kepada pasien pada lembaran-lembaran rekam medis dan menandatanganinya. Sementara itu pihak keluarga pasien melakukan pengisian proses registrasi pada bagian registrasi.
   d. Pasien sudah pulih dan diizinkan pulang oleh dokter, kemudian pihak pasien melakukan proses pembayaran.

      Berkas rekam medis pasien segera dikembalikan ke unit rekam medis. Kemudian petugas rekam medis mengolah rekam medis yang sudah lengkap dimasukkan ke dalam kartu indeks penyakit, indeks operasi,

indeks kematian dan sebagainya, untuk membuat laporan dan statistik rumah sakit.

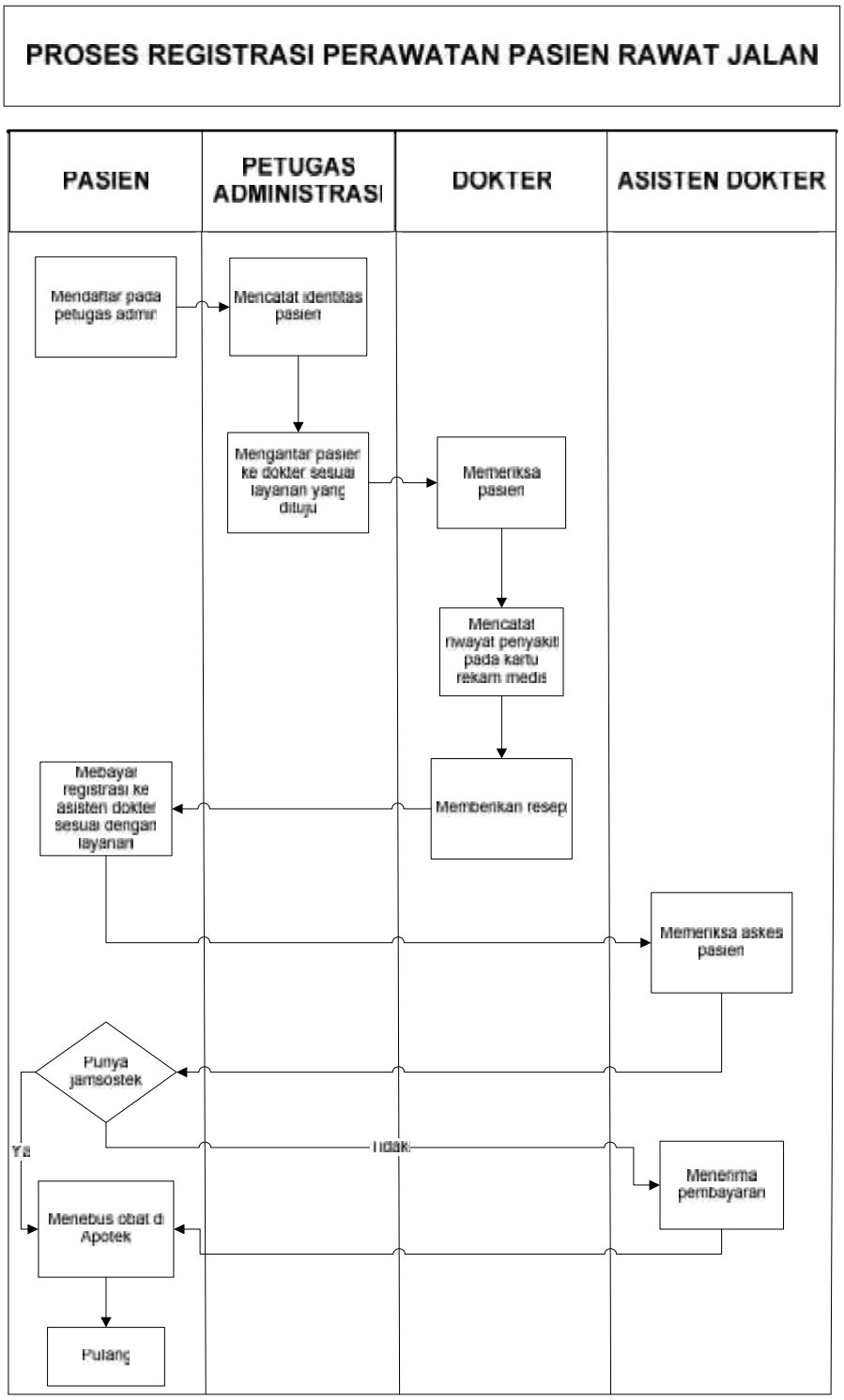


Gambar 4.1 Alur Pasien Rawat Inap

2. Proses registrasi perawatan pasien rawat jalan.

   Pada proses registrasi perawatan pasien rawat jalan ini, langkah-langkah yang dilalui oleh pasien adalah sebagai berikut :

   a. Pasien langsung mendaftar ketempat petugas bagian administrasi begitu juga untuk pasien yang mempunyai askes jamsostek, pasien jamsostek mendaftar untuk dilihat oleh petugas administrasi rekam medik pasien.
   b. Petugas bagian administrasi mencatat di buku pendaftaran : Nama pasien, No Rekam Medis, Identitas dan data sosial pasien dan mencatat keluhan pada kartu poliklinik. Bagi pasien rawat jalan yang lama, pasien cukup membawa kartu pasien. Kartu pasien berbentuk seperti kertas manila berukuran folio.
   c. Petugas bagian administrasi mengantar pasien keruang dokter tergantung jenis pelayanannya.
   d. Dokter memeriksa dan mencatat riwayat penyakit, hasil pemerikasaan dan diagnosis dengan penyakitnya pada kartu rekam medis pasien.
   e. Selesai di *check up* oleh dokter dan diberikan resep kemudian pasien membayar registrasi kepada asisten dokter sesuai dengan jenis pelayanannnya seperti pelayanan UGD 24 Jam, Poli Umum, Poli Kebidanan & Penyakit Kandungan, Poli Anak, KIA, Imunisasi, Konseling dan Pelayanan KB. Jika pasien melakukan pelayanan poli umum maka pasien langsung membayar kepelayanan poli umum.
   f. Resep obat diberikan kepasien dan ditebus keapotik RSIA Labuh Baru



Gambar 4.2 Alur Pasien Rawat Jalan

3. Proses Pembuatan Laporan

Pembuatan laporan yang berhubungan dengan pasien baik itu pasien rawat inap maupun rawat jalan sangat dibutuhkan dalam rangka menentukan langkah peningkatan pelayanan rumah sakit. Laporan-laporan ini umumnya berguna untuk pihak manajemen/pengelola rumah sakit serta pihak luar seperti Pemerintah Daerah dan pihak-pihak swata lainnya.

4.2 Analisa dan Perancangan Sistem Baru

Sistem diciptakan untuk membantu manusia memecahkan berbagai permaslahan, terutama masalah yang rumit dan jenis yang banyak. Analisa dan perancangan yang akan dibangun adalah analisa dan perancangan suatu sistem informasi yang memanfaatkan komputer sebagai perangkat utama pemrosesan. Manusia bertindak sebagai pengatur, pengoperasi, serta pengendali utama perangkat tersebut. Sistem berjalan setelah data masukan (*input*) diberikan. Data masukan (*input*) dalam sistem ini seperti Data Jabatan, Data Pegawai, Data Kamar, Data Fasilitas, Data Perawatan, Data Pelayanan, Data Pengguna, Data Tagihan dan Data ubah *password*. Setiap data pasien yang dimasukkan harus disimpan. Data pasien akan dimasukkan ke dalam sistem, maka sistem akan mengolah data pasien yang di*input*kan, sehingga keluar data akhir berupa data laporan pasien secara keseluruhan. Data ini akan diarsipakan di dalam *database* dan jika dibutuhkan sewaktu-waktu bisa dicari kembali dengan cepat dan akurat. Hasil yang diinginkan dari sistem ini adalah berupa informasi mengenai data laporan pasien rawat inap dan pasien rawat jalan RSIA Labuh Baru.

4.2.1 Data Masukan (*input*)

Data masukan (*input*) yang ada pada Sistem Informasi Perawatan Pasien RSIA Labuh Baru adalah:

1. Data jabatan pegawai memasukkan nama sub jabatan.
2. Data pegawai memasukkan nama Id pegawai.
3. Data kamar memasukkan nama kamar dan harga kamar.

4. Data fasilitas memasukkan nama fasilitas dan harga fasilitas.
5. Data perawatan memasukkan nama perawatan dan harga perawatan.
6. Data pelayanan memasukkan nama pelayanan dan harga pelayanan.
7. Data pengguna memasukkan nama *user, user name, password* dan level.
8. Data tagihan memasukkan data tagihan.
9. Data ubah *password* memasukkan data *user name* dan *password* baru.

4.2.2 Proses

Proses yang terjadi pada analisa dan perancangan adalah proses penyimpanan data pegawai, data jabatan pegawai, data kamar, data fasilitas, data perawatan, data pelayanan, data pengguna, data tagihan, data pasien dan laporan pasien data pegawai. Penyimpanan terjadi saat pengguna memasukkan data master, data tagihan dan data ubah *password* yang telah dimasukkan akan disimpan di *database* dalam sistem. Apabila dibutuhkan, arsip ini dapat dilihat kembali dengan mudah dan cepat. Pada laporan, proses ini dilakukan untuk menghasilkan informasi tentang data-data laporan pegawai beserta laporan pasien rawat inap dan pasien rawat jalan.

4.2.3 Data Keluaran (*output*)

Bentuk keluaran (*output*) yang akan ditampilkan sistem ini adalah berupa:

1. Informasi data pegawai.
2. Informasi data pasien.

Berdasarkan dari analisa kebutuhan dan spesifikasi sistem diatas dapat dilakukan perancangan Sistem Informasi Perawatan Pasien Rumah Sakit Ibu dan Anak (SIPPRSIA) dengan menggunakan UML (*Unified Modeling Language*).

UML adalah merupakan standar dalam menentukan,visualisasi, konstruksi, dan mendokumentasikan *artifact* dari sistem *software*, untuk memodelkan bisnis, dan sistem *nonsoftware* lainnya.

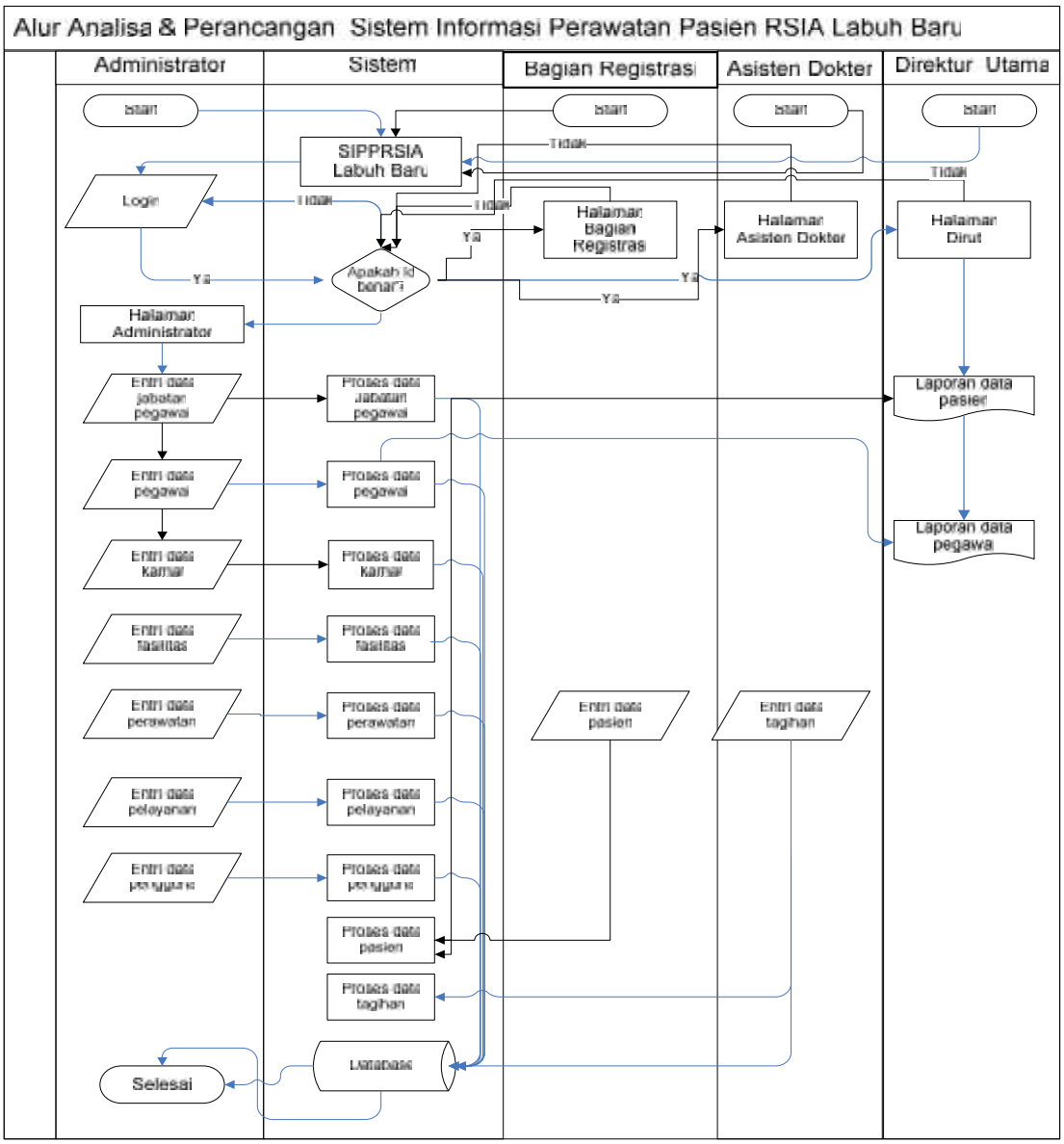
Model analisa dan perancangan SIPPRSIA ini sebagai berikut :

1. *Use Case Diagram.*
2. *Class Diagram.*
3. *Sequence Diagram.*

4. Deployment Diagram.

4.2.4 Analisa *Flowchart* Sistem

Untuk memperjelas proses yang terjadi pada SIPPRSIA ini, dapat digambarkan dengan menggunakan *flowchart* sebagai berikut:



Gambar 4.3 *Flowchart* SIPPRSIA Labuh Baru

4.2.5 *Use Case Diagram*

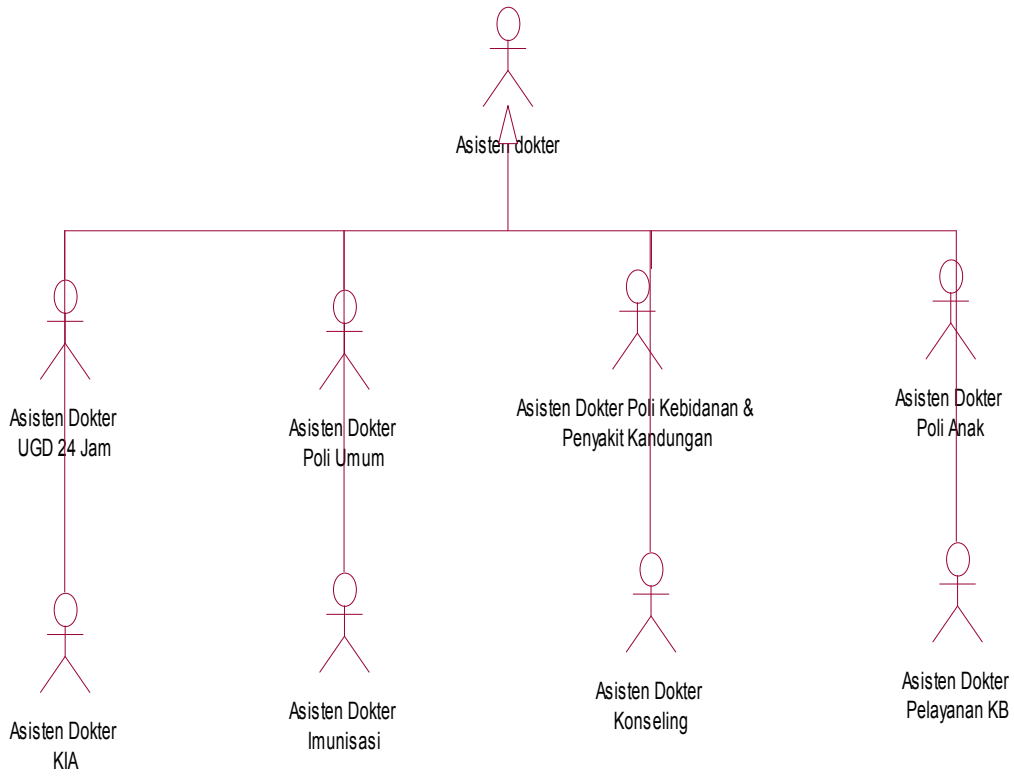
Use case diagram terdiri dari *actor, use case* dan beserta hubungannya.

Diagram *use case* adalah sesuatu yang penting untuk memvisualisasikan, menspesifikasikan, dan mendokumentasikan kebutuhan prilaku sistem.

4.2.5.1 *Actor*

Ada 4 *actor* yang terdapat dalam SIPPRSIA yaitu Administrator, Bagian Registrasi, Asisten Dokter dan Direktur Utama RSIA Labuh Baru. Karena dari masing-masing *actor* ini mempunyai hak akses pengguna yang berbeda. Adminstrator bertugas hanya mengelola sistem dari SIPPRSIA, Bagian Registrasi bertugas menginputkan data pasien rawat jalan dan pasien rawat inap yang akan melakukan pemeriksaan atau menggunakan jasa pelayanan RSIA, Asisten Dokter bertugas menginputkan data tagihan pasien karena di RSIA Labuh Baru ini asisten dokter pengganti dari kasir yang masing-masing pelayanan mempunyai masing-masing asisten dokter dan Direktur Utama RSIA Labuh Baru hanya bertugas melihat laporan harian dari laporan pegawai dan pasien.

Asisten Dokter generalisasi dari Asisten Dokter UGD 24 Jam, Asisten Dokter Poli Umum, Asisten Dokter Poli Kebidanan & Penyakit Kandungan, Asisten Dokter Poli Anak, Asisten Dokter KIA, Asisten Dokter Imunisasi, Asisten Dokter Konseling, dan Asisten Dokter Pelayanan KB. Pewarisan (*inheritance*) menggambarkan generalisasi sebuah kelas. Pewarisan merupakan hubungan antara class dimana dalam satu class ada super*class* atau *class* induk dari *class* yang lain. Pewarisan menunjuk pada properti dan *behaviour* yang diterima dari nenek moyang dari *class*. Seperti gambar dibawah ini merupakan generalisasi dari asisten dokter yaitu:



Gambar 4.4 *Actors* SIPPRSIA

Tabel 4.1 Deskripsi *Actors* SIPPRSIA Labuh Baru

Actor	**Deskripsi**
Administrator	Mengelola hak akses system
Bagian Registrasi	Mengelola pendaftaran pasien.
Asisten Dokter	1. Asisten Dokter UGD 24 Jam: melayani pasien yang ketika membutuhkan pertolongan secara mendadak atau yang datang pada malam hari, pasien tersebut akan diberi pelayanan sementara diruang UGD 24 Jam. 2. Asisten Dokter Poli Umum: melayani pasien diluar asuransi jamsostek. 3. Asisten Dokter Kebidanan & Penyakit Kandungan: melayani pasien tentang seputar pelayanan bidan dan segala sesuatu

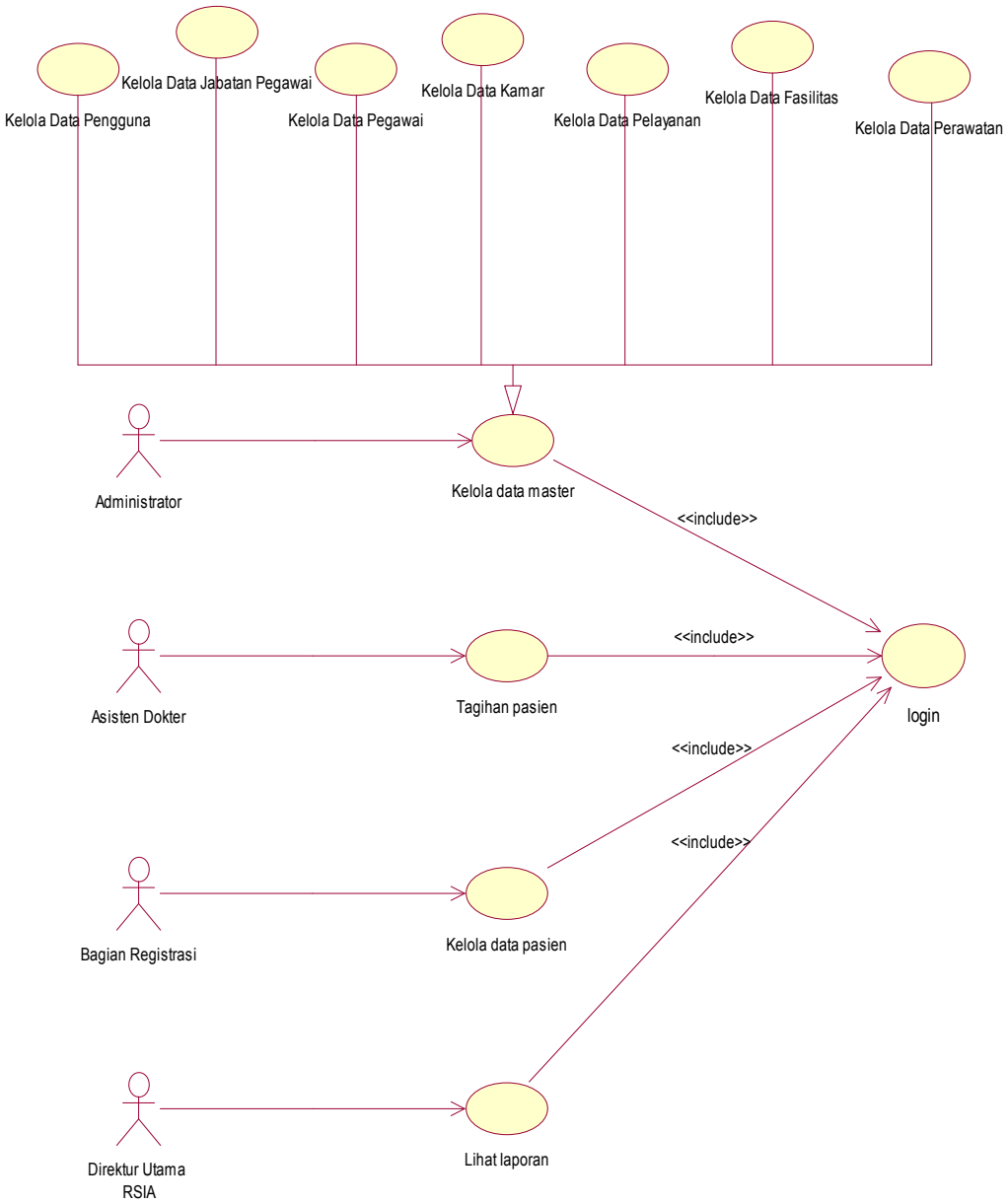
	tentang penyakit kandungan 4. Asisten Dokter Poli Anak: melayani pasien khusus penyakit anak. 5. Asisten Dokter KIA: melayani pasien khusus tentang kesehatan ibu dan anak. 6. Asisten Dokter Imunisasi: melayani pasien khusus tentang imunisasi. 7. Asisten Dokter Konseling: melayani pasien yang merangkup semuanya terutama pasien yang berkonsultasi tentang pembayaran bagi pasien yang kurang mampu. 8. Asisten Dokter Pelayanan KB: melayani pasien khusus dalam melayani segala macam jenis KB. a. Pelayanan KB Pil b. Pelayanan KB Suntik c. Pelayanan KB Spiral d. Pelayanan KB Susuk e. Pelayanan KB Steril
Direktur Utama	Melihat semua laporan pasien

4.2.5.2 Analisa *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng*create* sebuah daftar belanja, dan sebagainya. Seorang aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di*include* akan dipanggil setiap kali *use case* yang meng*include* dieksekusi secara normal. Sebuah *use case* dapat di*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common* . Sebuah *use case* juga dapat meng*extend use case* lain dengan *behaviour*nya sendiri. Sementara

hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Adapun secara umum *use case diagram* yang terdapat pada SIPPRSIA adalah sebagai berikut:



Gambar 4.5 *Use case* diagram SIPPRSIA

Tabel 4.2 berikut menjelaskan tentang *use case* diagram SIPPRSIA Labuh Baru

Tabel 4.2 *Spesification Use Case* Diagram *Verifikasi User* SIPPRSIA Labuh Baru

Aktor Utama	Administrator
Aktor Lain	-
Kondisi Awal	User belum *login*
Kondisi Akhir	User *log out*
Main Succes Scenario	1. Use case dimulai ketika mengklik menu login 2. User menginputkan *user name* dan *password* *3.* Sistem memverifikasi *user name* dan *password* 4. Sistem membiarkan user masuk
Alternate Scenario	3.1 Jika *user name* dan *password* salah maka sistem tidak bias membiarkan user masuk. 3.2 *User* diminta memasukkan kembali *user name* dan *password* dengan benar.

Spesifiaksi *use case* diagram *verifikasi user* dideskripsikan berdasarkan beberapa atribut spesifikasi seperti yang terlihat pada tabel 4.2. Prekondisi menunjukkan bahwa kondisi awal *user* belum login.

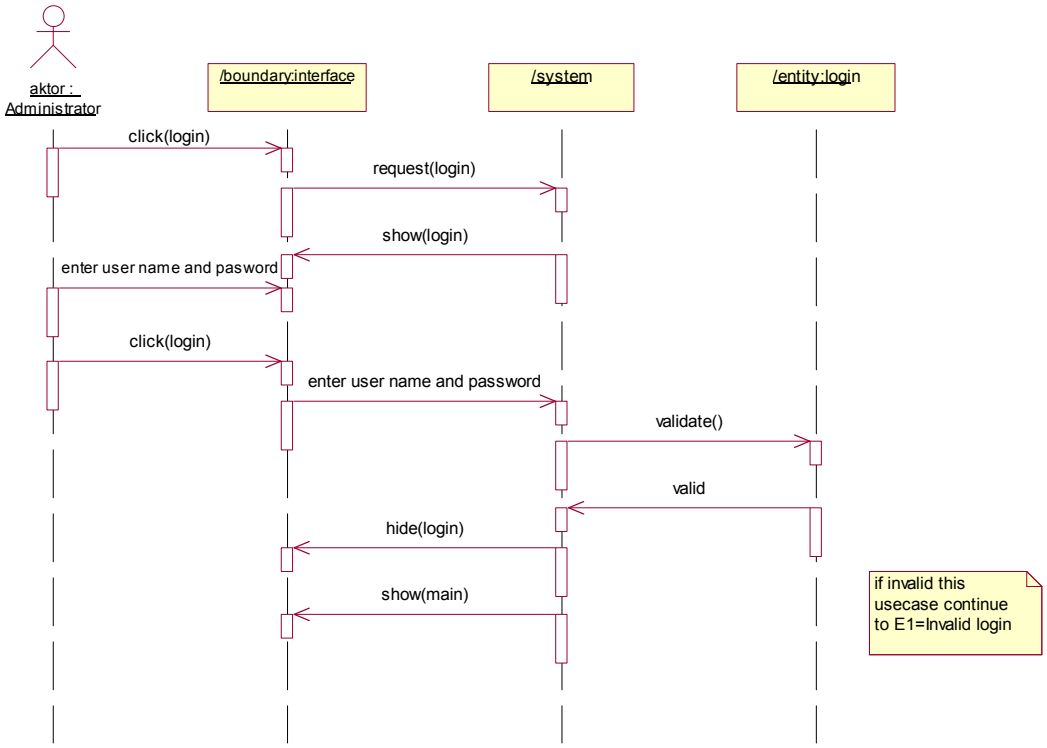
Deskripsi *usecase* diagram untuk proses-proses lain secara lengkap dapat dilihat pada lampiran A.

4.2.5.3 Analisa *Sequence Diagram*

Sequence diagram mendokumentasikan komunikasi/interaksi antar kelas-kelas. *Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem termasuk *user* berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah kasus untuk menghasilkan *output* tertentu. Diawali dari apa yang menyebabkan aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki garis vertikal. Pesan digambarkan sebagai garis berpanah dari satu objek ke

objek lainnya. Pada fase desain berikutnya, pesan akan dipetakan menjadi operasi/*method* dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah pesan.

Gambar 4.5 berikut menjabarkan *sequence diagram* SIPPRSIA Labuh Baru



Gambar 4.6 *Sequence Diagram valid login*

Sequence diagram diatas menunjukkan bahwa *sequence diagram valid login* berinteraksi dengan tiga *class* yaitu *interface*, sistem, dan *entity login*. Pemrosesan diawali dengan pemanggilan form menu dengan menunggunakan *method request*(). Selanjutnya sistem akan melakuakn validasi akun dengan menggunakan *method validate*() dan menampilkan menu utama. Pada tahapan terakhir kotak login akan tersembunyikan dan user dapat mengakses menu utama.

Sequence diagram untuk proses-proses lain secara lengkap dapat dilhat pada lampiran B.

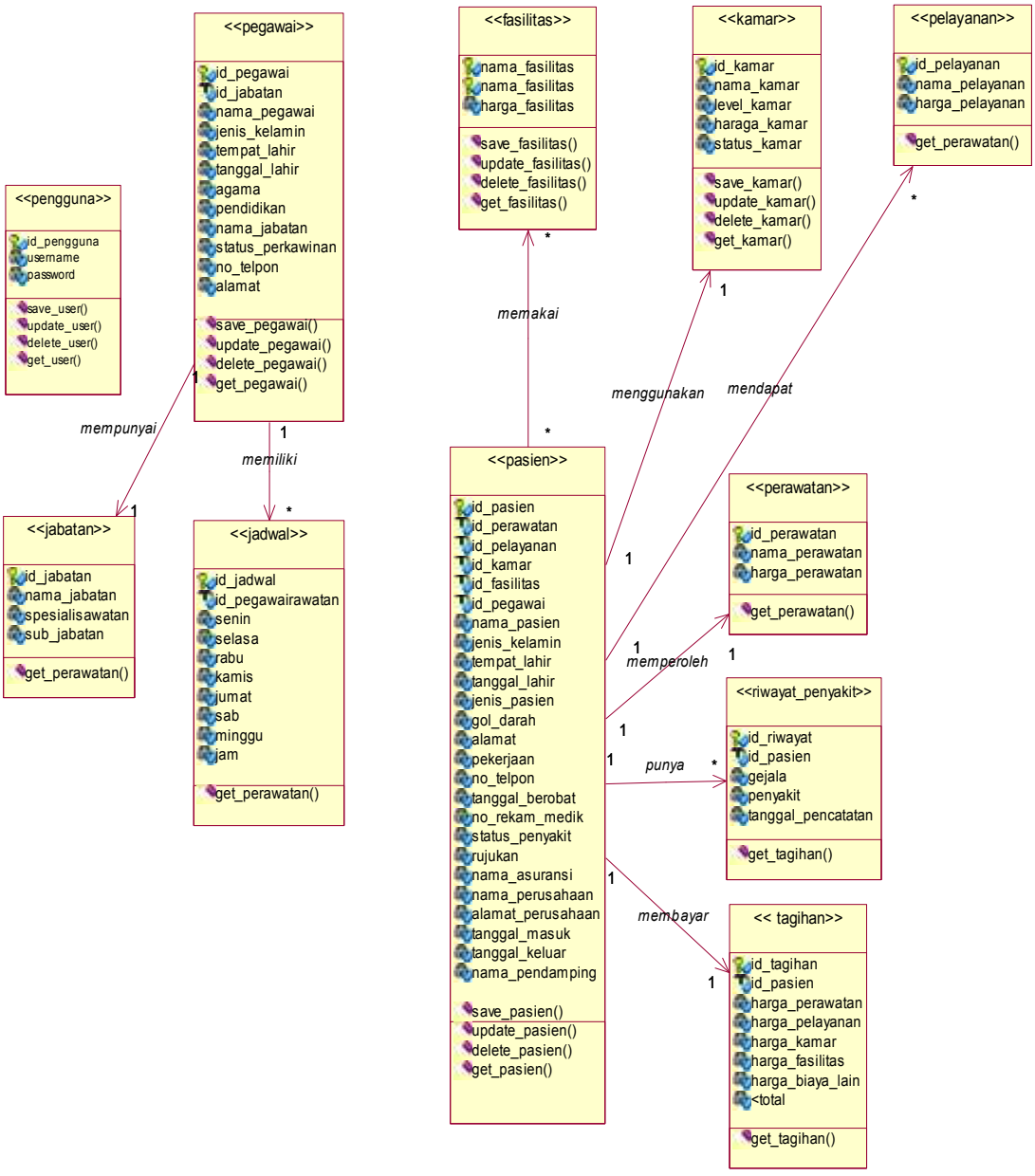
4.2.5.4 Analisa *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut yang disebut dengan *method. Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment* , pewarisan, asosiasi, dan lain-lain.
Class memiliki tiga area pokok yaitu: nama, atribut dan *method*.

Class dapat merupakan implementasi dari sebuah *interface* , yaitu *class* abstrak yang hanya memiliki *method*. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi *method* pada saat *runtime* .
Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package* . Kita juga dapat membuat diagram yang terdiri atas *package*.
Hubungan Antar *Class*

1. Asosiasi yaitu hubungan statis antar *class* . Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability m* enunjukkan arah *query* antar *class* .
2. Agregasi yaitu hubungan yang menyatakan bagian (“terdiri atas”).
3. Pewarisan yaitu hubungan hirarkis antar *class* . *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya.
4. Hubungan dinamis yaitu rangkaian pesan yang di *passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

Hasil perancangan Gambar 4.7 dibawah ini menjelaskan kebutuhan *class diagram* SIPPRSIA Labuh Baru.



Gambar 4.7*Class Diagram*

Dari gambar diatas terlihat bahwa 8 terdapat *class* yaitu *class user*, class pegawai, *class* pasien, *class* kamar, *class* fasilitas, *class* perawatan dan *class* tagihan. Tabel berikut ini menjelaskan ketarangan *class diagram*.

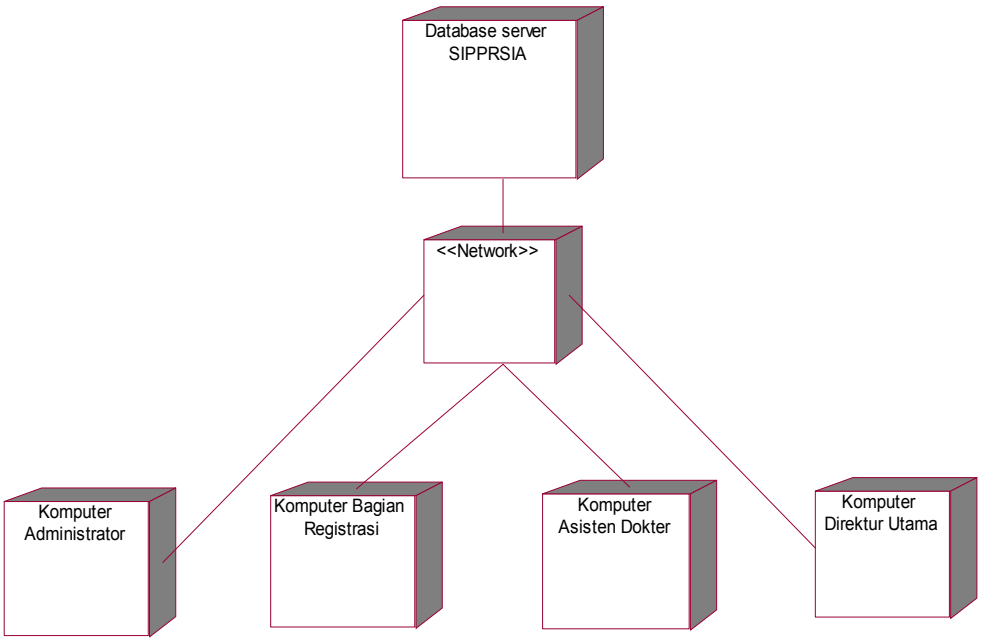
Tabel 4.3 Keterangan *Class Diagram*

No	Nama *Class*	Keterangan
1	Pengguna	Merupakan kelas pengguna
2	Pasien	Merupakan kelas yang berhubungan dengan data pasien
3	Jabatan	Merupakan kelas yang berhubungan dengan data jabatan pegawai
4	Pegawai	Merupakan kelas yang berhubungan dengan data pegawai
5	Pelayanan	Merupakan kelas yang berhubungan dengan data pelayanan
6	Fasilitas	Merupakan kelas yang berhubungan dengan data fasilitas
7	Kamar	Merupakan kelas yang berhubungan dengan data kamar
8	Perawatan	Merupakan kelas yang berhubungan dengan data perawatan
9	Jadwal Dokter	Merupakan kelas yang berhubungan dengan data jadwal dokter
10	Riwayat Penyakit	Merupakan kelas yang berhubungan dengan data riwayat penyakit pasien
11	Tagihan	Merupakan kelas yang berhubungan dengan data tagihan

4.2.5.5 Analisa *Deployment Diagram*

Deployment/physical diagram menjelaskan hubungan antara *software* dan *hardware* yang ada serta menggambarkan detail bagaimana komponen di sebarkan dalam infrastruktur sistem, di mana komponen akan terletak pada mesin, server bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah *server*, *workstation* , atau piranti keras lain yang digunakan untuk menyebarkan komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

Hasil analisa dan perancangan gambar 4.8 dibawah ini menggambarkan kebutuhan *deployment diagram* yang menjelaskan hubungan antara *software* dan *hardware* yang digunakan di dalam analisa dan perancangan sistem informasi perawatan pasien RSIA Labuh Baru.



Gambar 4.8 *Deployment Diagram*

Gambar analisa dan perancangan *deployment diagram* perawatan pasien RSIA Labuh Baru diatas menggambarkan hubungan antara komputer *client* dengan *server*, yang terdiri dari *database server* SIPPRSIA yang saling berhubungan dengan komputer administrator, komputer bagian registrasi, komputer asisten dokter dan komputer direktur utama yang dihubungkan oleh *Local Area Network* (LAN).

4.3 Penerapan Aspek

Aspek didenifisikan sebagai properti yang merentang pada sekelompok komponen fungsional. Sementara aspek-aspek tersebut dapat dianalisis secara relatif terpisah dari fungsionalitas dasar. Pada level implementasi aspek-aspek harus dikombinasikan bersama. *Aspect Oriented Programing* (AOP) dirancang untuk menangani masalah lintas sektoral. AOP tidak mengganti bahasa paradigma pemrograman yang ada melainkan bekerja dengan *Object Oriented Programming* (OOP) untuk meningkatkan kegunaan OOP.

Aspek pengkapsulannya terpisah tetapi saling berhubungan satu dengan yang lainnya dan pada *class diagram* harus mempunyai *get save* di *method.* Sedangkan pada Objek *Oriented* malah sebaliknya.

Berikut adalah penerapan aspek yang terjadi pada perawatan pasien rawat inap dan rawat jalan pada RSIA Labuh Baru adalah :

4.3.1 Analisa dan Perancangan Aspek Yang Mempengaruhi Aktifitas Pasien Rawat Jalan.

Analisa aspek yang mempengaruhi aktifitas rawat jalan adalah:

Tabel 4.4 Keterangan aspek rawat jalan

	Pasien	Pelayanan	Total Tagihan
Registrasi Pasien	SavePasien() UpdatePasien()	SavePelayanan() UpdatePelayanan()	
Patients in Check	DeletePasien()	DeletePelayanan()	
Patients Out	GetPasien()	GetPelayanan()	GetTagihan()

Pasien	Pelayanan	Total Tagihan
*nama_pasien	nama_pelayanan	id_pasien
id_pasien	*id_pelayanan	id_pelayanan
id_pelayanan	id_fasilitas	id_pegawai
id_kamar	id_kamar	nama_tagihan
id_fasilitas	harga_pelayanan	harga_kamar
id_pegawai	level_kamar	harga_pelayanan
id_perawatan		harga_fasilitas
nama_pegawai		harga_perawatan
jenis_perawatan		*total_tagihan
no_rekam_medik		
status_penyakit		
gol_darah		
Alamat		
TTL		
jenis_kelamin		
Pekerjaan		
no_telp		
Rujukan		

tgl_masuk tgl_keluar ket_pasien harga_fasilitas harga_pelayanan harga_kamar nama_pendamping_pasien status_pendamping_pasien total_tagihan		

Keterangan :

* aspek

1. Sebelum menerima layanan kesehatan pasien mendaftar terlebih dahulu ke bagian registrasi, apakah pasien tersebut pasien rawat jalan atau pasien rawat inap kemudian administrator menginputkan nama pasien.
2. Pasien menyampaikan pelayanan yang akan mereka butuhkan sesuai dengan keperluan pasien tersebut.
3. Setelah selesai *check up*, pasien membayar total pembayaran kepada asisten dokter sesuai dengan poli pelayanan pasien.

Proses diatas harus saling berhubungan satu sama lain, seperti kelas pasien, kelas pelayanan dan kelas tagihan. Jika salah satunya tidak ada kegiatan yang saling mengisi maka sangat mempengaruhi sistem dan objek harus terpenuhi.

Tabel 4.4 menunjukkan bahwa analisa aspek dilihat secara fungsional. Maksud dari fungsional adalah dilihat dari sudut pandang masalah yang terjadi didalam RSIA Labuh Baru tersebut. Masalah yang diangkat adalah mengenai perawatan pasien rawat inap dan rawat jalan. Fungsional harus dilihat secara garis horizontal yaitu dengan cara memotong kelas dalam sistem (ditampilkan sebagai kotak) dan setiap kelas harus mempunyai *method.* Setiap kotak daftar operasi yang dikenakan pada kelas masing-masing. *Source Code* yang sedehana untuk pemeriksaan pasien rawat jalan menunjukkan daftar disederhanakan yang mengimplementasikan fungsi *Patiens Out* sebagai aspek menggunakan AspectJ.

Sourcode untuk *Patients Out*:

```
1.  public aspect Patients Out {
2.    ...
3.    public void Pasien.getPasien ()
4.    {
5.      // code to Patients Out
6.    }
7.    public void Pelayanan.getPelayanan ()
8.    {
9.      // code to getPelayanan
10.   }
11.   public void Total Tagihan.getTagihan ()
12.   {
13.     // code to generate an initial outstanding tagihan
14.   }
15.   ...
16. }
```

Baris 1 pada *Sourcode Listing* untuk *Patients Out* menyatakan bahwa fungsionalitas *Patients Out* sebagai aspek seperti ditunjukkan oleh kata kunci aspek bahwa fungsional dilihat secara garis horizontal yang berisi serangkaian *intertype declarations* (kemungkinan untuk menulis fitur baru seperti atribut, *method*, dan hubungan kelas-kelas yang ada. Ada dua segmen dari sebuah *intertype declarations* yaitu segmen pertama adalah nama dari sebuah kelas yang ada dan segmen kedua adalah operasi yang sudah ada yang ingin ditambahkan ke dalam baris ke 3. Contoh misalnya:

```
Pasien.getPasien()
```

Menambahkan operasi ( ) *getPelayanan* keruang kelas Pasien. Perhatikan bahwa meskipun operasi ini adalah bagian dari kelas Pasien, didefinisikan di luar kelas pasien itu sendiri. Jadi orientasi aspek telah lolos dari pembatasan pada *method* kelas yang perlu didefinisikan dalam modularitas kelas tradisional.

4.3.2 Analisa dan Perancangan Aspek Yang Mempengaruhi Aktifitas Pasien Rawat Inap.

Analisa aspek yang mempengaruhi aktifitas rawat inap adalah:

Tabel 4.5 Keterangan analisa aspek rawat inap

	Pasien	Pelayanan	Kamar	Total Tagihan
Registrasi Patient	SavePasien()	SavePelayanan()	SaveKamar()	
Reserve Room	UpdatePasien()	UpdatePelayanan()	UpdateKamar()	
Check In Patient	DeletePasien()	DeletePelayanan()	DeleteKamar()	
Check Out Patient	GetPasien()	GetPelayanan()	GetKamar()	GetTagihan()

Pasien	Pelayanan	Kamar	Total Tagihan
*nama_pasien	nama_pelayanan	nama_kamar	id_pasien
id_pasien	*id_pelayanan	id_kamar	id_pelayanan
id_pelayanan	id_fasilitas	*level_kamar	id_pegawai
id_kamar	id_kamar	status_kamar	nama_tagihan
id_fasilitas	harga_pelayanan	harga/nett	harga_kamar
id_pegawai	level_kamar		harga_pelayanan
id_perawatan			harga_fasilitas
nama_pegawai			harga_perawatan
jenis_perawatan			*total_tagihan
no_rekam_medik			
status_penyakit			
gol_darah			
Alamat			
TTL			
jenis_kelamin			
Pekerjaan			
no_telp			
Rujukan			
tgl_masuk			
tgl_keluar			
ket_pasien			
harga_fasilitas			
harga_pelayanan			
harga_kamar			
nama_pendamping_pasien			
status_pendamping_pasien			
total_tagihan			

Keterangan :

* aspek

Analisa aspek yang mempengaruhi aktifitas rawat inap adalah kelas pasien, kelas pelayanan, kelas kamar dan kelas tagihan.

1. Pasien meregistrasi kepada bagian administrasi bahwa objeknya adalah pasien rawat inap lalu bagian registrasi meng*input*kan nama pasien.
2. Jika pasien tersebut menggunakan pelayanan maka di*input*kan nama pelayanannya.
3. Untuk penyediaan kamar, administrator memeriksa di sistem apakah ada kamar yang tersedia atau penuh.

   Jika ada kamar yang kosong administrator membuat membuat pemesanan tempat dengan sesuai levelnya.
4. Pasien menuju ke kamar yang kosong, pada waktu yang sama administrator membuat data tagihan rawat inap.
5. Pasien *check out*, administrator mengecek total pembayaran dari asisten dokter sesuai dengan lamanya pasien memakai pelayanan.

Sourcode untuk *Check Out Patient*:

```
1.  public aspect Check Out Patient {
2.     ...
3.     public void Pasien.getPasien ()
4.     {
5.       // code to Check Out Patient
6.     }
7.     public void Pelayanan.getPelayanan ()
8.     {
9.       // code to getPelayanan
10.    }
11.    public void Kamar.getKamar ()
12.    {
13.      // code to getKamar
14.    }
15.    Public void Total Tagihan.getTagihan ()
16.    {
17.      // code to generate an initial outstanding tagihan
18.    }
15.    ...
16. }
```

Baris 1 pada *Sourcode Listing* untuk *Check Out Patient* menyatakan bahwa fungsionalitas *Check Out Patient* sebagai aspek seperti ditunjukkan oleh kata kunci aspek bahwa fungsional dilihat secara garis horizontal yang berisi serangkaian *intertype declarations* (kemungkinan untuk menulis fitur baru seperti atribut, *method*, dan hubungan kelas-kelas yang ada. Ada dua segmen dari sebuah *intertype declarations* yaitu segmen pertama adalah nama dari sebuah kelas yang ada dan segmen kedua adalah operasi yang sudah ada yang ingin ditambahkan ke dalam baris ke 3. Contoh misalnya:

```
Pasien.getKamar()
```

Menambahkan operasi ( ) *getKamar* keruang kelas Pasien. Perhatikan bahwa meskipun operasi ini adalah bagian dari kelas Pasien, didefinisikan di luar kelas pasien itu sendiri. Jadi orientasi aspek telah lolos dari pembatasan pada *method* kelas yang perlu didefinisikan dalam modularitas kelas tradisional.

4.3.3 *Pseudocode* Perawatan Pasien Rawat Inap dan Rawat Jalan Menggunakan Pendekatan Berorientasi Aspek *Oriented*

Pseudocode untuk Analisa & Perancangan Sistem Informasi Perawatan Pasien RSIA Labuh Baru dalam aspek *oriented* pada perawatan pasien rawat jalan dan rawat inap adalah:

Pseudocode 1 (aspek *oriented*)

```
1. a,n,RI,RJ,getPasien(),getPelayanan(),getKamar(),getTagihan()
2. a = n
3. input → RI
4.        if n←RI then Check Out Patient
5.              if Check Out Patient←true than
6.          input→getPasien(),getPelayanan(),getKamar(),getTagihan()
7.        and
8.          output→RI
9.           else if
10.          RI→ false
11.   end if
```

```
12.   RJ
13.          if Patients Out→false then
14.               input→getPasien(),getPelayanan(),getTagihan() and
15.               output→RJ
16.          end if
17.          RI→false
18.   output→RJ or
19.   output→RI
20.end.
```

4.3.4 *Pseudocode* Perawatan Pasien Rawat Inap dan Rawat Jalan Menggunakan Pendekatan Berorientasi Objek *Oriented*

Pseudocode untuk Analisa & Perancangan Sistem Informasi Perawatan Pasien RSIA Labuh Baru dalam objek *oriented* pada perawatan pasien rawat jalan dan rawat inap adalah:

Pseudocode 2 (objek *oriented*)

```
1. a, n, RI, RJ, d-kam, d-par, d-pel, d-perwtn, d-pas
2.    a = n
3.    input → RI
4.          if d-kam ← true than
5.            input→ d-pas
6.            input→d-pel
7.            input→d-perwtn
8.             input→d-kamar
9.             input→d-fas
8.            input→d-tagihan and
9.            output→RI
10.         else if
11.   RI→ false
12.   end if
13.   RJ
14.         If d-kam→false then
```

```
15.             input→d-pas
16.             input→d-pel
17.             input→d-perwtn
18.             input→d-tagihan and
19.             output→RJ
20.          end if
21.          RI→false
22.   output→RJ or
23.   output→RI
24. end.
```

Ket:

RI: Rawat Inap

RJ: Rawat Jalan

d-kam: data kamar

d-pel: data pelayanan

d-perwtn: data perawatan

d-fas: data fasilitas.

4.3.5 Perbedaan *Pseudocode* Pendekatan Berorientasi Aspek *Oriented* & Berorientasi Objek *Oriented* Dalam Perawatan Pasien Rawat Inap dan Pasien Rawat Jalan

Perbedaan yang terletak pada *Pseudocode* 1 (aspek *oriented*) dan *Pseudocode* 2 (objek *oriented*) adalah:

Pendekatan Berorientasi Aspek *Oriented* untuk proses perawatan pasien Rawat Inap pada *Pseudocode* 1 yaitu aspek terlihat dari operasi no 4, dimana proses Rawat Inap terdapat pada operasi *Check Out Patients*. Pada proses no 6, terdapat kelas-kelas yang terdiri dari empat kelas yaitu kelas pasien, kelas pelayanan, kelas kamar dan kelas total tagihan dan masing–masing kelas dipanggil melalui empat *method* yaitu : *method getPasien*(), *getPelayanan* ( ), *getKamar* ( ) dan *getTagihan* ( ) yang saling berhubungan satu sama lainnya.

Pendekatan Berorientasi Objek *Oriented* untuk proses perawatan pasien Rawat Inap pada *Pseudocode* 2 proses Rawat Inap yaitu objek terlihat pada operasi no 5 sampai operasi no 8 dimana proses Rawat Inap masing-masing kelas dipanggil satu per satu pada kelas. Misalnya pada proses pemanggilan kelas kamar maka pemanggilan *code* nya dari kelas kamar dan begitu seterusnya.

Pendekatan Berorientasi Aspek *Oriented* untuk proses perawatan pasien Rawat Jalan pada *Pseudocode* 1 yaitu aspek terlihat pada operasi no 13, dimana proses Rawat Jalan terdapat pada operasi *Patients Out*. Pada proses no 14, terdapat kelas-kelas yang terdiri dari tiga kelas yaitu kelas pasien, kelas pelayanan dan kelas total tagihan dan masing–masing kelas dipanggil melalui tiga *method* yaitu : *method getPasien*(), *getPelayanan* ( ) dan *getTagihan* ( ) yang *method* nya saling berhubungan satu sama lainnya. Inilah yang dinamakan aspek karena proses *crosscutting concern* (proses pemotongan *code* yang dibaca secara horizontal).

Pendekatan Berorientasi Objek *Oriented* untuk proses perawatan pasien Rawat Jalan pada *Pseudocode* 2 yaitu objek terlihat pada operasi no 15 sampai operasi no 18 dimana proses Rawat Jalan masing-masing kelas dipanggil satu per satu pada kelas. Misalnya pada proses pemanggilan kelas perawatan maka pemanggilan *code* nya dari kelas perawatan dan begitu seterusnya.

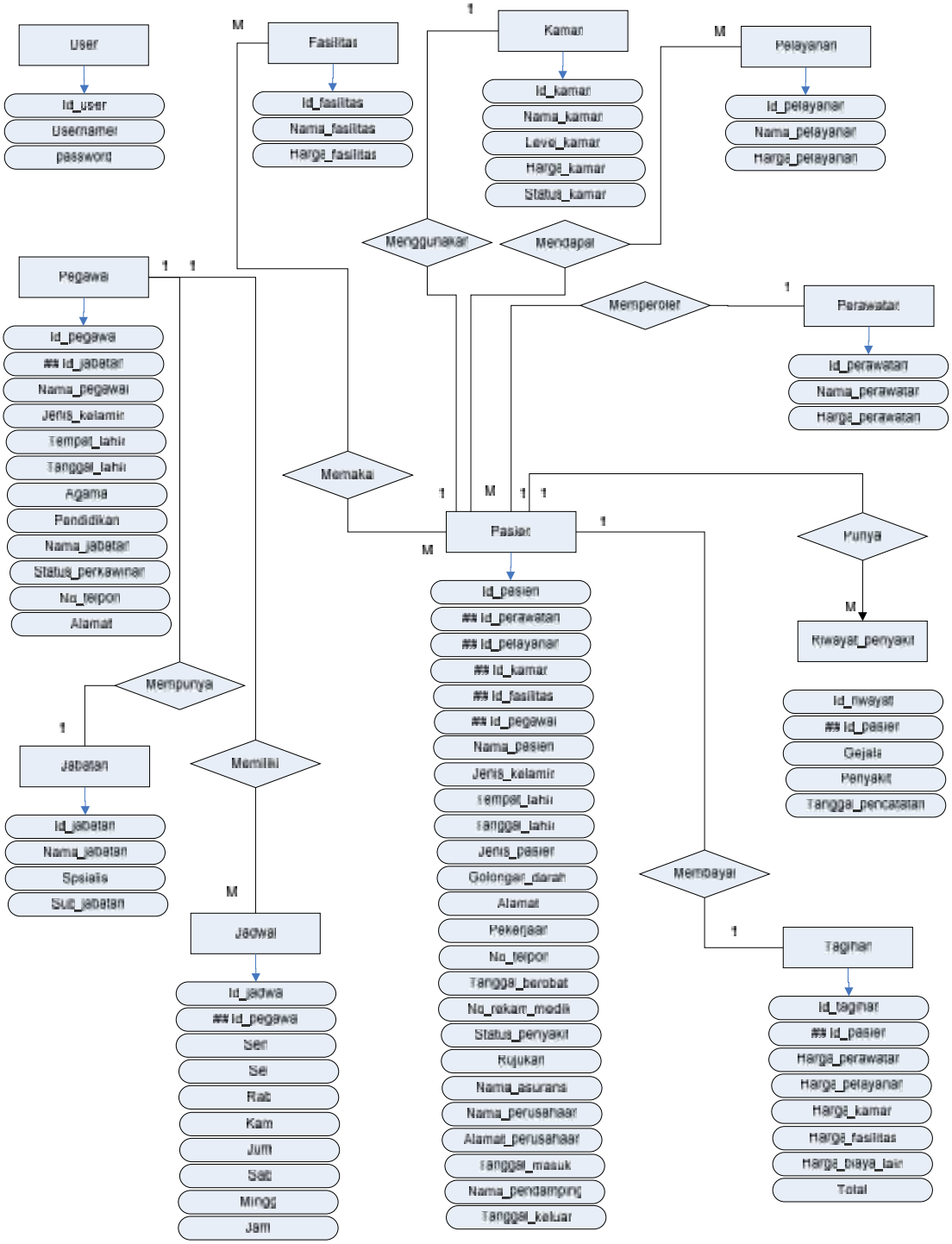
4.4 Perancangan

Perancangan objek dalam konsep *Object Oriented Database* terutama adalah penghalusan serta penambahan rincian. Langkah-langkah perancangan objek adalah:

1. Merancang akses ke basis data, yang meliputi langkah-langkah sebagai berikut:
   a. Organisasi basis data secara fisik dan logika
   b. Pemetaan model objek ke tabel-tabel basisdata
2. Merancang antarmuka pengguna yang interaktif, yang meliputi langkah-langkah sebagai berikut:
   a. Merancang kelas-kelas pada tampilan yang hadir di hadapan pengguna
   b. Menentukan kegunaan antarmuka interaktif.

4.4.1 Perancangan Basisdata

Gambar 3.7 adalah model basisdata dari Sistem Informasi Perawatan Pasien Rumah Sakit Ibu dan Anak Labuh Baru.



Gambar 4.9 Model Basisdata SIPPRSIA Labuh Baru

Tabel 4.6 Keterangan Atribut Tabel Pengguna

no	Nama field	Tipe data	Null	Keterangan
1	Id_*user*	Integer(3)	Not null	Id *user*
2	Nama_*user*	Varchar(50)	Not null	Nama *user*
3	*Username*	Varchar(50)	Not null	*Username*
4	*Password*	Varchar(50)	Not null	*Password user*
5	Level	Varchar(15)	Not null	Level_*user*

Tabel 4.7 Keterangan Atribut Tabel Pelayanan

no	Nama field	Tipe data	Null	Keterangan
1	Id_pelayanan	Integer(3)	Not null	Id pelayanan
2	Nama_pelayanan	Varchar(100)	Not null	Nama pelayanan
3	Harga_pelayanan	Integer (10)	Not null	Harga pelayanan

Tabel 4.8 Keterangan Atribut Tabel Fasilitas

no	Nama field	Tipe data	Null	Keterangan
1	Id_fasilitas	Integer(3)	Not null	Id fasilitas
2	Nama_fasilitas	Varchar(50)	Not null	Nama fasilitas
3	Harga_fasilitas	Integer(10)	Not null	Harga fasilitas

Tabel 4.9 Keterangan Atribut Tabel Pasien

no	Nama field	Tipe data	Null	Keterangan
1	Id_pasien	Varchar(20)	Not null	Id pasien
2	Id_pelayanan	Integer(3)	Not null	Id pelayanan
3	Nama_pasien	Varchar(50)	Not null	Nama pasien
4	Jk	Char(2)	Not null	Jenis kelamin pasien
5	Tempat_lahir	Varchar(50)	Not null	Tempat lahir pasien
6	Tanggal	Varchar(50)	Not null	Tanggal lahir pasien
7	Jenis_pasien	Varchar(20)	Not null	Jenis pasien
8	Golongan_darah	Char(2)	Not null	Golongan darah pasien
9	Alamat	Text	Not null	Alamat pasien
10	Pekerjaan	Varchar(40)	Not null	Pekerjaan pasien
11	No_telpon	Varchar(13)	Not null	No telpon pasien
12	Id_perawatan	Integer(3)	Not null	Id perawatan pasien
13	Id_kamar	Integer(3)	Not null	Id kamar pasien
14	Id_fasilitas	Integer(3)	Not null	Id fasiltas pasien
15	Tanggal_berobat	Date	Not null	Tanggal berobat pasien
16	No_rekam_medik	Varchar(50)	Not null	No rekam medik pasien

17	Status_penyakit	Varchar(20)	Not null	Status penyakit pasien
18	Id_pegawai	Varchar(20)	Not null	Id pegawai
19	Rujukan	Varchar(100)	Not null	Rujukan pasien
20	Nama_asuransi	Varchar(50)	Not null	Nama asuransi pasien
21	Nama_perusahaan	Varchar(50)	Not null	Nama perusahaan pasien
22	Alamat_perusahaan	Text	Not null	Alamat perusahaan_pasien
23	Tanggal_masuk	Varchar(50)	Not null	Tanggal masuk pasien
24	Nama_pendamping	Varchar(50)	Not null	Nama pendamping pasien
25	Tanggal_keluar	Varchar(50)	Not null	Tanggal keluar pasien

Tabel 4.10 Keterangan Atribut Tabel Pegawai

no	Nama field	Tipe data	Null	Keterangan
1	Id_pegawai	Varchar(10)	Not null	Id pegawai
2	Nama_pegawai	Varchar(50)	Not null	Nama pegawai
3	Jk	Varchar(5)	Not null	Jenis kelamin pegawai
4	Tempat_lahir	Varchar(50)	Not null	Tempat lahir pegawai
5	Tanggal	Varchar(30)	Not null	Tanggal lahir pegawai
6	Agama	Varchar(15)	Not null	Agama pegawai
7	Pendidikan	Varchar(4)	Not null	Pendidikan pegawai
8	Nama_jabatan	Varchar(100)	Not null	Nama jabatan pegawai
9	Id_jabatan	Integer(3)	Not null	Id jabatan pegawai
10	Status_perkawinan	Varchar(20)	Not null	Status perkawinan pegawai
11	No_telpon	Varchar(12)	Not null	No telpon pegawai
12	Alamat	Text	Not null	Alamat

Tabel 4.11 Keterangan Atribut Tabel Tagihan

no	Nama field	Tipe data	Null	Keterangan
1	Id_tagihan	Integer(3)	Not null	Id tagihan
2	Id_pasien	Varchar(10)	Not null	Id pasien
3	Harga_perawatan	Integer(10)	Not null	Harga perawatan
4	Harga_pelayanan	Integer(10)	Not null	Harga pelayanan
5	Harga_kamar	Integer(10)	Not null	Harga kamar
6	Harga_fasilitas	Integer(10))	Not null	Harga fasilitas
7	Harga_biaya_lain	Integer(10)	Not null	Harga biaya lain
8	Total	Integer(10)	Not null	Total

Tabel 4.12 Keterangan Atribut Tabel Kamar

no	Nama field	Tipe data	Null	Keterangan
1	Id_kamar	Integer(3)	Not null	Id kamar
2	Nama_kamar	Varchar(25)	Not null	Nama kamar
3	Level_kamar	Varchar(25)	Not null	Level kamar
4	Harga_kamar	Integer(10)	Not null	Harga kamar

5	Status_kamar	Varchar(10)	Not null	Status kamar

Tabel 4.13 Keterangan Atribut Tabel Jabatan

no	Nama field	Tipe data	Null	Keterangan
1	Id_jabatan	Integer(3)	Not null	Id jabatan pegawai
2	Nama_jabatan	Varchar(25)	Not null	Nama jabatan pegawai
3	Spesialis	Varchar(30)	Not null	Sesialis pegawai
4	Sub_jabatan	Varchar(50)	Not null	Sub jabatan pegawai

Tabel 4.14 Keterangan Atribut Tabel Jadwal

no	Nama field	Tipe data	Null	Keterangan
1	Id_jadwal	Integer(3)	Not null	Id jadwal dokter
2	Id_dokter	Varchar(10)	Not null	Id dokter
3	Hari	Varchar(10)	Not null	Hari kerja
4	Jam	Varchar(30)	Not null	Jam kerja

Tabel 4.15 Keterangan Atribut Tabel Perawatan

no	Nama field	Tipe data	Null	Keterangan
1	Id_perawatan	Integer(3)	Not null	Id perawatan
2	Nama_ perawatan	Varchar(100)	Not null	Nama perawatan
3	Harga_ perawatan	Integer (10)	Not null	Harga perawatan

Tabel 4.16 Keterangan Atribut Tabel Riwayat Penyakit

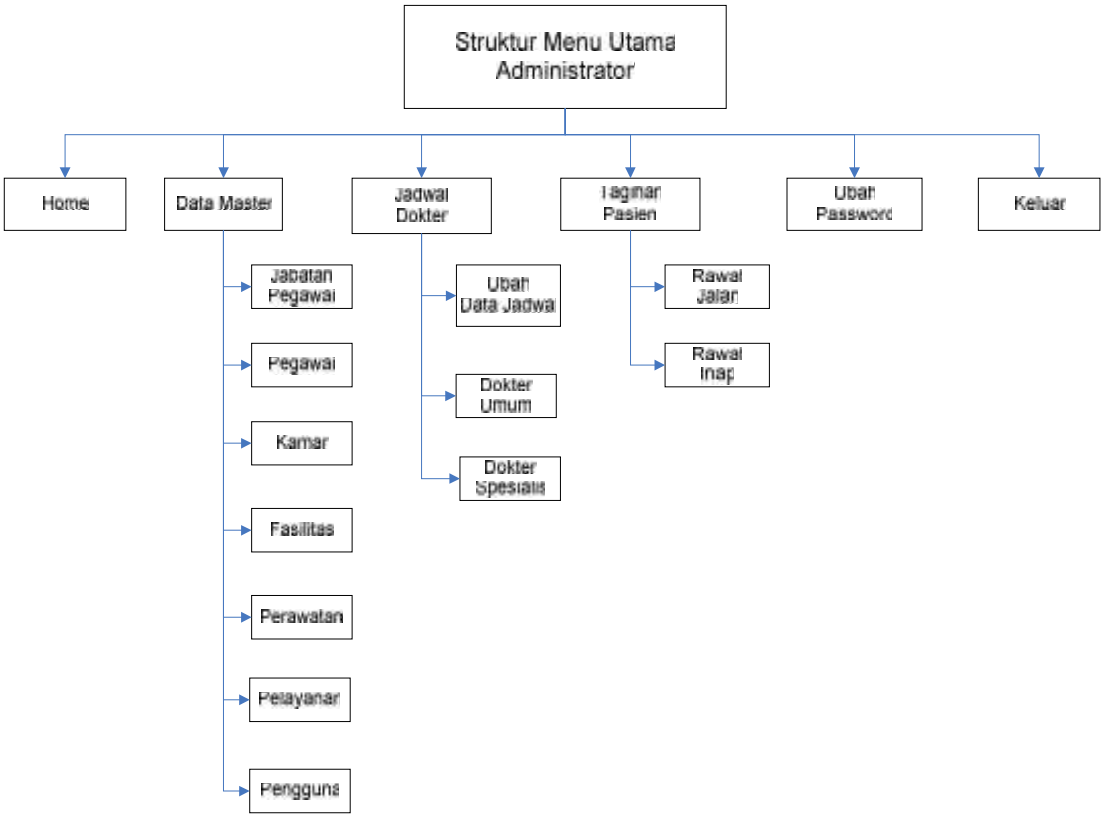
no	Nama field	Tipe data	Null	Keterangan
1	Id_riwayat	Integer(3)	Not null	Id riwayat
2	Id_pasien	Varchar(20)	Not null	Id pasien
3	Gejala	Text	Not null	Gejala
4	Penyakit	Text	Not null	Penyakit
5	Tanggal_pencatatan	Varchar(50)	Not null	Tanggal pencatatan

4.4.2 Perancangan Subsistem Dialog

Merancang subsistem dialog berupa tampilan menu sistem yang *user friendly* sehingga *user* paham dalam menggunakan atau memilih menu-menu pilihan terdapat pada sistem.

4.4.2.1 Struktur Menu

Berikut ini merupakan gambar struktur menu SIPPRSIA. Sistem ini terdiri 4 level hak akses pengguna yang terdiri dari administrator, bagian registrasi,asisten dokter dan direktur utama RSIA. Struktur menu administrator terdiri dari dua menu dan beberapa menu memiliki sub-sub menu. Struktur menu sistem setelah melakukan login administrator pada Sistem Informasi Perawatan Pasien RSIA Labuh Baru dapat dilihat pada gambar 4.10 berikut ini.

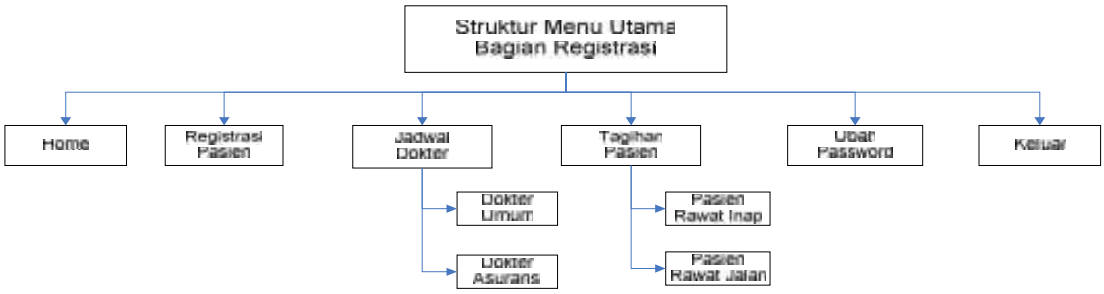


Gambar 4.10 Struktur Menu Sistem Setelah Login Administrator

Tabel 4.17 Deskripsi Struktur Menu Setelah Login Administrator

No	Menu	Menu Item	Fungsi
1	Home	-	Tampilan awal setelah melakukan *login* pengguna
2	Data Master	Jabatan Pegawai	Input data jabatan pegawai yang dilengkapi

			dengan tambah, ubah dan hapus
3	Data Master	Pegawai	Input data pegawai yang dilengkapi dengan tambah, ubah dan hapus
4	Data Master	Kamar	Input data kamar yang dilengkapi dengan tambah, ubah dan hapus
5	Data Master	Fasilitas	Input data fasilitas yang dilengkapi dengan tambah, ubah dan hapus
6	Data Master	Perawatan	Input data perawatan yang dilengkapi dengan tambah, ubah dan hapus
7	Data Master	Pelayanan	Input data pelayanan yang dilengkapi dengan tambah, ubah dan hapus
8	Data Master	Pengguna	Input data pengguna yang dilengkapi dengan tambah, ubah dan hapus
9	Jadwal Dokter	Ubah Data Jadwal	Mengubah data jadwal dokter
10	Jadwal Dokter	Dokter Umum	Input data jadwal dokter umum dari data master pegawai yang dilengkapi dengan ubah dan hapus
11	Jadwal Dokter	Dokter Spesialis	Input data jadwal dokter spesialis dari data master yang dilengkapi dengan ubah dan hapus
12	Tagihan Pasien	Rawat Jalan	Input tagihan pasien rawat jalan yang dilengkapi dengan cetak dan hapus
13	Tagihan Pasien	Rawat Inap	Input tagihan pasien rawat inap yang dilengkapi dengan cetak dan hapus
14	Ubah Password	-	Proses ubah password *login* pengguna
15	Keluar	-	Proses keluar aplikasi dari hak akses administrator



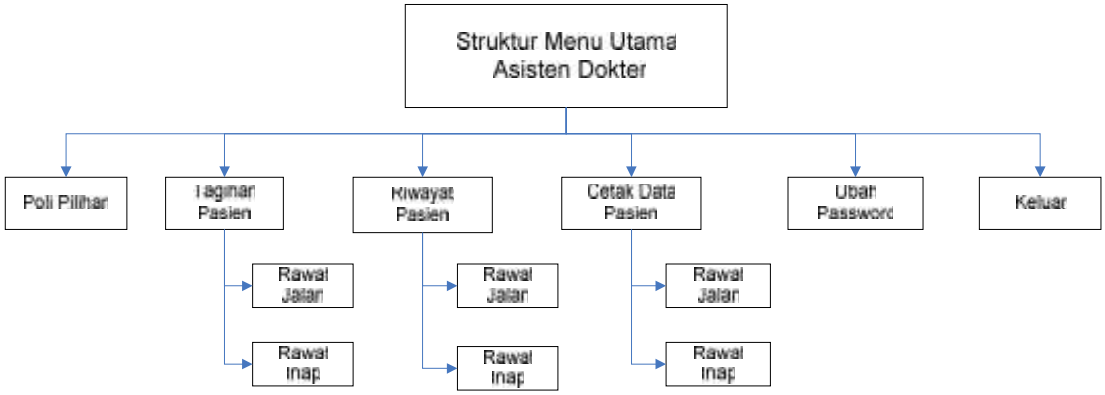
Gambar 4.11 Struktur Menu Sistem Setelah Login Bagian Registrasi

Tabel 4.18 Deskripsi Struktur Menu Setelah Login Bagian Registrasi

No	Menu	Menu Item	Fungsi
1	Home	-	Tampilan awal setelah melakukan *login*

			pengguna
2	Registrasi Pasien	-	Input data registrasi pasien yang dilengkapi dengan simpan data pasien
3	Jadwal Dokter	Dokter Umum	Informasi jadwal dokter umum yang hanya bisa di*input* dari data master pegawai.
4	Jadwal Dokter	Dokter Spesialis	Informasi data jadwal dokter spesialis yang hanya bisa di*input* dari data master pegawai.
5	Tagihan Pasien	Pasien Rawat Inap	Informasi tagihan pasien rawat inap yang hanya bisa di*input* dari hak akses level asisten dokter.
6	Tagihan Pasien	Pasien Rawat Jalan	Input tagihan pasien rawat jalan yang hanya bisa di*input* dari hak akses level asisten dokter.
7	Ubah Password	-	Proses ubah password *login* pengguna
8	Keluar	-	Proses keluar aplikasi dai hak akses bagian registrasi

Pada gambar 4.12 berikut merupakan struktur menu setelah melakukan login pada level asisten dokter. Admin berhasil masuk ke sistem aplikasi. Pada *form* ini ada beberapa menu utama, berikut struktur menu utama asisten dokter



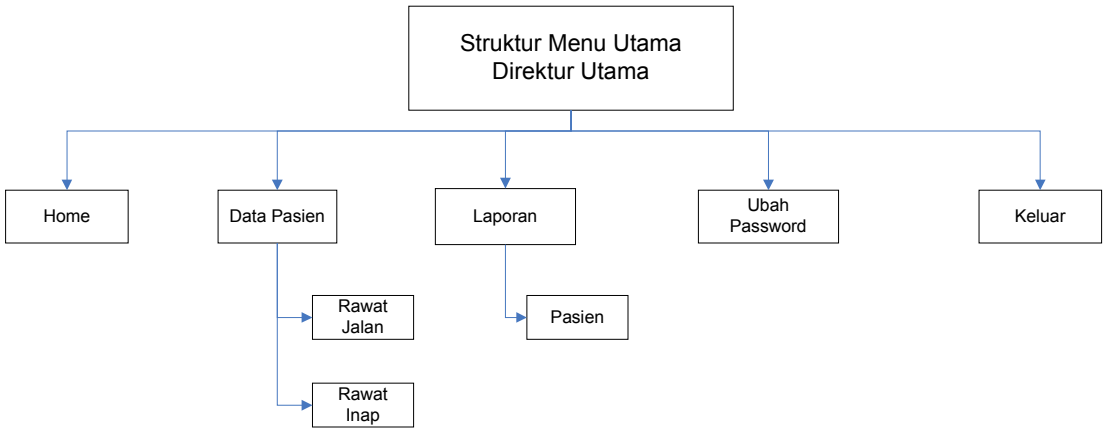
Gambar 4.12 Struktur Menu Sistem Setelah Login Level Asisten Dokter

Tabel 4.19 Deskripsi Struktur Menu Setelah Login Level Asisten Dokter

No	Menu	Menu Item	Fungsi
1	Poli Pilihan	-	Pilih menu poli yang sesuai dengan kebutuhan pasien

2	Tagihan Pasien	Rawat Jalan	Input data tagihan pasien rawat jalan yang dilengkapi dengan menu cetak.
3	Tagihan Pasien	Rawat Inap	Input data tagihan pasien rawat inap yang dilengkapi dengan menu cetak.
4	Riwayat Pasien	Rawat Jalan	Input data riwayat pasien rawat jalan yang dilengkapi dengan menu cetak.
5	Riwayat Pasien	Rawat Inap	Input data riwayat pasien rawat inap yang dilengkapi dengan menu cetak.
6	Cetak Data Pasien	Rawat Jalan	Cetak data pasien rawat jalan
7	Cetak Data Pasien	Rawat Inap	Cetak data pasien rawat inap
8	Ubah Password	-	Proses ubah password *login* pengguna
9	Keluar	-	Proses keluar aplikasi dai hak akses asisten dokter

Pada gambar 4.13 berikut *form* ini ada beberapa menu utama, berikut struktur menu utama direktur.



Gambar 4.13 Struktur Menu Sistem Setelah Login Level Direktur Utama

Tabel 4.20 Deskripsi Struktur Menu Setelah Login Level Direktur Utama

No	Menu	Menu Item	Fungsi
1	Home	-	Tampilan awal setelah melakukan *login* pengguna
2	Data Pasien	Rawat Jalan	Laporan data pasien khusus rawat jalan
3	Data Pasien	Rawat Inap	Laporan data pasien khusus rawat jalan
4	Laporan	Pegawai	Laporan data pegawai RSIA Labuh Baru
5	Laporan	Pasien	Laporan data riwayat penyakit pasien beserta total tagihan
6	Ubah Password	-	Proses ubah password *login* pengguna
7	Keluar	-	Proses keluar aplikasi dari hak akses direktur

			utama

4.4.2.2 Perancangan Antarmuka

Pada subbab ini akan ditampilkan rancangan antar muka sistem informasi perawatan pasien RSIA Labuh Baru.

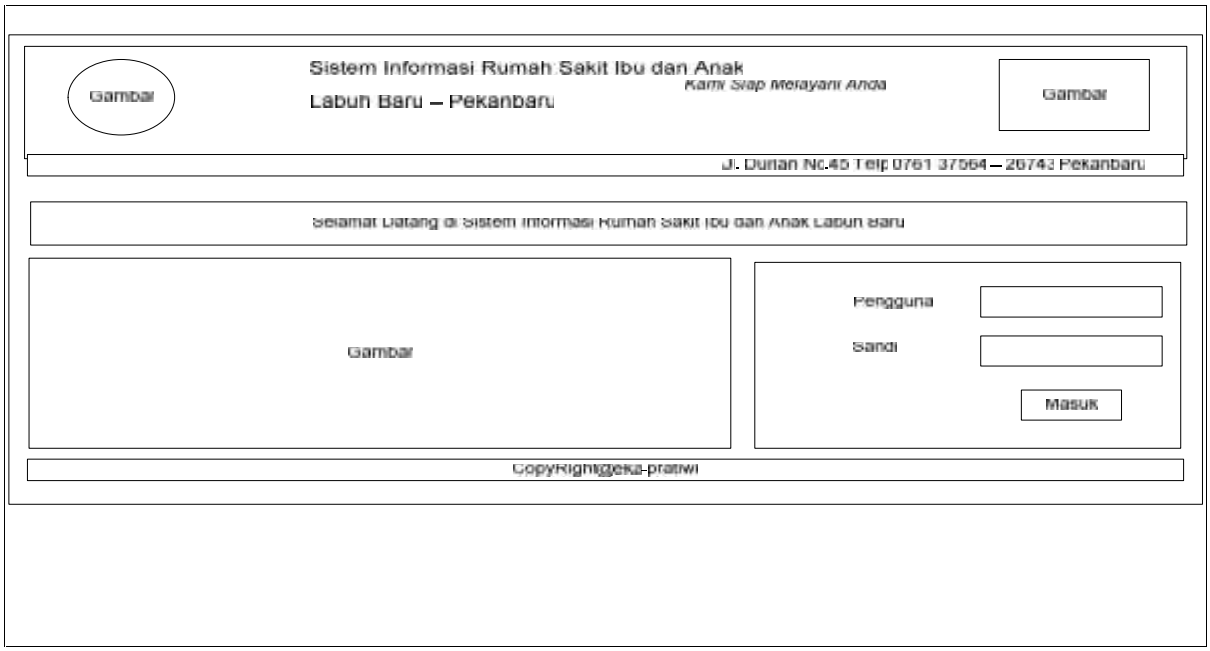
Rancangan antar muka sistem adalah sarana pengembangan sistem yang digunakan untuk membuat komunikasi yang baik dan konsisten antara sistem dengan pemakainya. Penekanan rancangan antar muka meliputi tampilan yang baik, mudah dipahami, tombol-tombol yang *familiar* serta *user friendly*.

Sistem pada level administrator memiliki beberapa antar muka yaitu *home*, data master, jadwal dokter, tagihan pasien, ubah *password* dan keluar. Sistem pada level bagian registrasi memiliki beberapa antar muka yaitu *home*, registrasi pasien, jadwal dokter, tagihan pasien, ubah *password* dan keluar sistem. Sistem pada level asisten dokter memiliki beberapa antar muka yaitu poli pilihan, tagihan pasien, riwayat pasien, cetak data pasien, ubah *password* dan keluar sistem. Sedangkan sistem pada level direktur utama memiliki beberapa antar muka yaitu *home*, data pasien, laporan, ubah *password* dan keluar sistem.

Yang berhak untuk mengakses *form* ini adalah pengguna dengan level administrator, pengguna dengan level bagian registrasi dan pengguna level asisten dokter. *Form-form* ini berfungsi untuk untuk menginputkan data pasien serta tagihan-tagihan pasien. Sedangkan pada pengguna dirut(direktur utama) hanya bisa melihat laporan data pasien di web sistem.

4.4.2.2.1 Perancangan *Form Login*

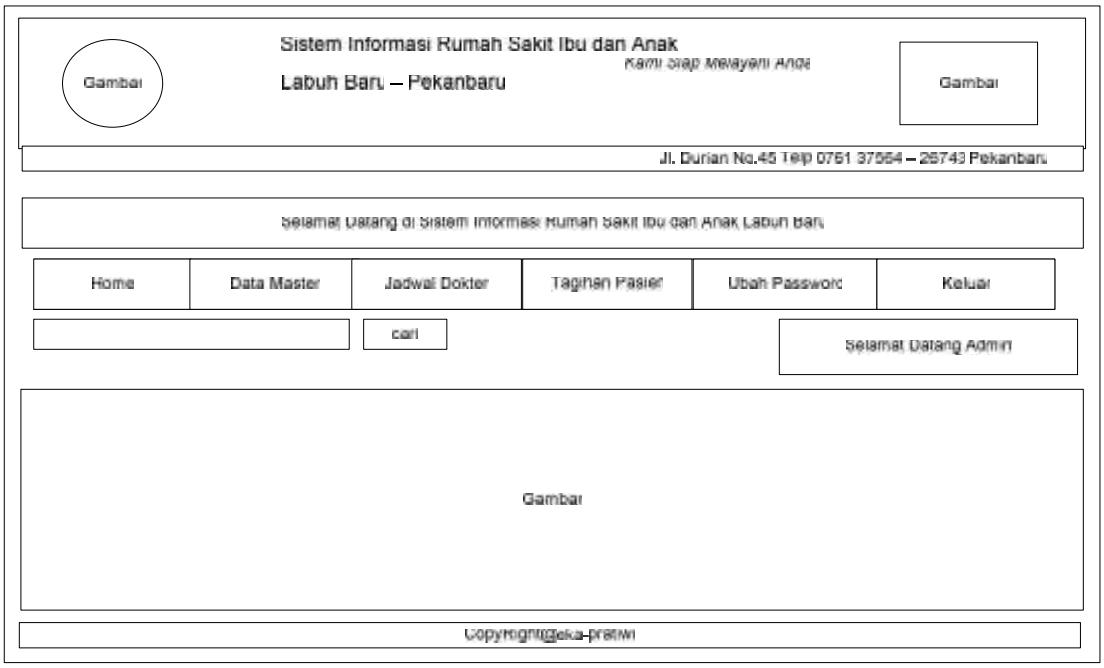
Berikut ini adalah perancangan *form login,* yang terdiri dari pengguna dan sandi. *Form* ini akan muncul pada saat pertama kali program dijalankan dengan memasukkan data pengguna dan sandi dengan benar. Pada bagian ini jika proses *login* berhasil maka sistem akan masuk kedalam menu administrator dan melakukan *update* aplikasi dan sebaliknya jika *login* gagal maka sistem akan kembali ke menu *login* ini dengan beberapa otentikasi dan peringatan yang telah dibuat.



Gambar 4.14 Perancangan *Form Login*

4.4.2.2.2 Perancangan *Form* Utama

Menu utama akan tampil dihalaman *administrator* RSIA, apabila proses login berhasil setelah memasukkan data pengguna dan sandi dengan benar, administrator dapat mengakses menu-menu yang terdapat dalam sistem tersebut untuk mengelola *content website* dan fitur kegiatan RSIA yang terdiri dari *home*, data master, jadwal dokter, tagihan pasien, ubah *password* dan keluar.



Gambar 4.15 Perancangan *Form* Utama

Pada Gambar 4.15 akan dijelaskan antarmuka *form* tambah master dan selanjutnya akan dijelaskan secara rinci pada lampiran C.

BAB V
IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

Implementasi merupakan tahap dimana sistem siap diopersikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat benar-benar dapat menghasilkan tujuan yang ingin dicapai. Pada tahap ini difokuskan kepada penerapan sistem yang didesain pada bahasa pemrograman yang sesuai, sehingga akan diperoleh hasil yang akan diinginkan.

5.1.1. Tujuan Implementasi

Implementasi adalah kelanjutan dari tahap penyelesaian rancangan setelah didesain. Pada tahap ini menerapkan sistem yang akan didesain ke bahasa pemrograman yang sesuai sehingga diperoleh hasil yang diinginkan.

Adapun tujuan dari Implementasi Analisa & Perancangna Sistem Informasi Perawatan Pasien Pada RSIA Labuh Baru Berorientasi Aspek ini adalah:

1. Menyelesaiakan desain sistem yang ada dalam dokumentasi perancangan yang telah disetujui.
2. Menguji dan mendokumentasikan program-program atau prosedur-prosedur dari dokumen perancngan sistem yang telah disetujui.
3. Memastikan bahwa pemakai dapat mengoperasikan sistem yaitu dengan mempersiapkan secara manual pemakai serta melatih pemakai.
4. Mempertimbangkan bahwa sistem memenuhi permintaan pemakai yaitu dengan menguji secara keseluruhan.
5. Memastikan bahwa konversi kesistem baru berjalan dengan benar yaitu dengan membuat rencana, mengontrol dan melakukan instalasi sistem secara benar.

Langkah-langkah yang dibutuhkan dalam pengimplementasian sistem adalah sebagai berikut :

1. Menyelesaikan desain sistem
2. Mendapatkan hardware dan software yang sesuai
3. Menguji, mengontrol dan mendokumentasikan program komputer

4. Memilih dan melatih pemakai
5. Menguji sistem
6. Mendapatkan persetujuan.

5.1.2. Alasan Pemilihan Perangkat Lunak

Perangkat lunak yang digunakan dalam implementasi sistem informasi perawatan pasien pada RSIA Labuh Baru berorientasi aspek adalah PHP, *MySQL* version 5.0.27, *Mozila Firefox* 6.3 dan *Macromedia Dreamweaver* 8 dari adanya beberapa pertimbangan berikut :

1. *MySQL* mampu menangani data sangat besar.
2. Penggunaan database *MySQL* lebih *user friendly.*
3. PHP merupakan halaman web yang dinamis.
4. PHP dapat mengirim HTTP *header*, mengeset *cookies*, mengatur *authentication* dan *redirect users*.
5. PHP dan *MySQL* memiliki kecepatan dalam eksekusi perintah dan mampu menangani jutaan *request* secara bersamaan.
6. PHP menawarkan koneksitas yang baik dengan berbagai macam basis data.
7. PHP dan *MySQL* adalah *software* gratis dan mampu lintas *platform* yaitu dapat digunakan dengan sistem operasi dan *web server* apapun seperti *Windows* dan beberapa versi Linux.

5.1.3. Lingkungan Implementasi

Lingkungan implementasi adalah lingkungan dimana aplikasi ini dikembangkan. Lingkungan implementasi sistem ada dua yaitu *hardware*, yaitu kebutuhan perangkat keras computer dalam pengolahan data dan *software*, yaitu kebutuhan akan perangkat lunak berupa sistem untik mengoperasikan sistem yang telah didesain, dengan spesifikasi sebagai berikut:

1. Perangkat Keras

   Perangkat keras yang digunakan mempunyai spesifikasi sebagai berikut:

   a. *Processor* : Intel Pentium IV 2.4 GHz
   b. *Memory* : 512 MB
   c. *Hardisk* : 40 GB

2. Perangkat Lunak

   Perangkat lunak yang digunakan adalah sebagai berikut:

   1. *Operating System* : *Windows XP Professional*
   2. *Memory* : 512 MB
   3. Bahasa Pemrograman : PHP
   4. *Database* : MySQL

5.1.4. Batasan Implementasi

Dalam melakukan implementasi analisa dan perancangan sistem di berikan beberapa batasan-batasan tertentu supaya inti dari masalah tidak keluar dari jalurnya. Adapun batasan implementasi dari tugas akhir ini adalah :

1. Aplikasi ini dikembangkan dalam bentuk berbasis web dan berjalan pada komputer lokal (*localhost*)
2. Menggunakan bahasa pemograman PHP dan *Database* yang digunakan adalah MySQL.
3. Mengelola pencatatan data pasien rawat inap dan rawat jalan dengan mengunakan metode aspek.
4. Menu pada aplikasi bersifat dinamis artinya bisa menambah atau mengurangi menu yang sudah ada.

5.1.5 Teknis Implementasi

Teknis implementasi dilakukan dengan tahapan sebagai berikut:

1. Implementasi *login* SIPPRSIA (Sistem Informasi Perawatan Pasien Rumah Sakit Ibu dan Anak).

   Tahap ini di lakukan *login* pada *web portal*, ketika *user* admin menekan tombol *login* maka *user* admin akan di arahkan ke halaman beranda *administrator*. Disini *user* memasukkan *username* dan *password*.
2. Mengimplementasikan dan menguji SIPPRSIA pada lingkungan RSIA Labuh Baru.

5.1.6 Hasil Implementasi

Pemrograman merupakan kegiatan penulisan program yang akan dieksekusi oleh komputer berdasarkan hasil dari analisa dan perancangan sistem. Sebelum program diimplementasikan, maka program tersebut harus bebas dari

kesalahan. Pengujian program dilakukan untuk menemukan kesalahan-kesalahan yang mungkin terjadi.

5.1.6.1 Tampilan *Form* Login

Berikut ini adalah perancangan *form login,* yang terdiri dari *username, password*. *Form* ini akan muncul pada saat pertama kali program dijalankan dengan memasukkan data *Username* dan *Password* dengan benar. Setelah mengklik tombol *login*, sistem mengecek *database* dengan data *login* yang diinputkan oleh *user*, termasuk level hak akses *use*r dalam menggunakan sistem (level administrator, level bagian registrasi, level asisten dokter dan level direktur utama). Jika data yang diinputkan benar, akan masuk ke tampilan menu utama Tampilan menu *login* dapat dilihat pada gambar 5.1 di bawah ini.

Gambar 5.1 Tampilan *login*

Jika administrator salah memasukkan *username* dan *password* maka akan tampil form seperti berikut

Gambar 5.2 Tampilan *login salah*

5.1.6.2 Tampilan Menu Utama

Menu ini merupakan tampilan utama administrator dari Sistem Informasi Perawatan Pasien RSIA Labuh Baru. Tampilan menu utama merupakan sebuah tampilan yang pertama kali muncul ketika aplikasi dijalankan dengan mengetikkan alamat http://localhost/rsia/admin/index.php.

Tampilan menu utama dapat diakses jika menu *login* dinyatakan *valid* dan disesuaikan dengan level akses dari pengguna, yaitu sebagai Administrator, Bagian Registrasi, Asisten Dokter dan Direktur Utama.

1. Tampilan menu utama yang dapat diakses oleh administrator adalah menu *home*, data master, jadwal dokter, tagihan pasien, ubah *password* dan keluar.

Tampilan menu administrator dapat dilihat pada gambar 5.3 berikut ini.

Gambar 5.3 Tampilan Menu Utama Administrator

2. Tampilan menu utama yang dapat diakses oleh Bagian Registrasi terdiri *home*, registrasi pasien, jadwal dokter, tagihan pasien, ubah *password* dan keluar. Tampilan menu bagian registrasi dapat dilihat pada gambar 5.4 berikut ini.

Gambar 5.4 Tampilan Menu Utama Bagian Registrasi

3. Tampilan menu utama yang dapat diakses oleh Asisten Dokter terdiri poli pilihan, tagihan pasien, riwayat pasien, cetak data pasien, ubah *password* dan keluar. Tampilan menu asisten dokter dapat dilihat pada gambar 5.5 berikut ini.

Gambar 5.5 Tampilan Menu Utama Asisten Dokter

4. Tampilan menu utama yang dapat diakses oleh Direktur Utama terdiri *home*, data pasien, laporan, ubah *password* dan keluar. Tampilan menu direktur utama dapat dilihat pada gambar 5.6 berikut ini.

Gambar 5.6 Tampilan Menu Utama Direktur Utama

Untuk hasil Implementasi lebih rinci dapat dilihat pada lampiran D.

DAFTAR PUSTAKA

Adhari, "Pengantar Pemrograman Berorientasi Aspek", [Online] Available URL: http://www.ilmukomputer.com.

aop@groups.or.id.

Ari Yanuar Ridwan, "*mememperbaiki modularitas program dengan pemrograman berorientasi aspek*", Manajemen Informatika Politeknik Pos Indonesia, 2006.

Brown, Jeff. "*Developing Aspects with Spring AOP*". 2008.

Coad, Peter dan Yourdon. *Object-Oriented Analysis.* New York: Yourdon Press Prentice Hall Building, 1991.

Filman, R. "*Aspect-oriented programming is quantification and obliviousness*". Proceeding of OOPSLA 2000 workshop on Advanced Separation of Concerns, 2000.

Kristanto, Andi."*Sistem Informasi dan Aplikasinya*". Yokyakarta : Gava Media Yogyakarta, 2003.

Jacobson Ivan dan Pan-Wei Ng. "*Aspect-Oriented Software Development with Use Cases*", 2004.

Jogiyanto, H.M. "*Analisa dan Desain Sistem Informasi*". Yogyakarta : Andi Offset, 1999.

Kristanto, Andri. "*Perancangan Sistem Informasi dan Aplikasinya*". Jakarta: Penerbit Gava Media, 2003.

Machacek, Jan., Vukotic, Aleksa., Chakraborty, Anirvan., & Ditt, Jessica. Pro Spring 2.5, United States of America: Appress, 2008.

Nugroho, Adi. "*Analisis dan Perancangan Sistem Informasi dengan Metodologi Berorientasi Objek"*. Bandung : Informatika, 2005.

Suhendar, A. "*Visual Modelling menggunakan UML dan Relational Rose*". Bandung: Penerbit Informatika Bandung, 2002.

Sutopo, Ariesto Hadi. "*Analisis dan Desain Berorientasi Objek*". Jakarta: Penerbit J & J Learning, 2002.

http://iaprima.staff.gunadarma.ac.id/Downloads/files/5456/Bahasan7_UML_bagian1.

http://www.digituck.com/konsep-oop-object-oriented-programming.html.